AD-A100 803 AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOO--ETC F/G 9/2
A GENERALIZED EXPERIMENT CONTROL AND DATA ACQUISITION SYSTEM.(U)
MAR 81 A R ILORETA
UNCLASSIFIED AFIT/GCS/EE/81M-3 NL

1 OF 1
AD A
100803

AFIT/GCS/EE/81M-3

Accession For	
NTIS GRA&I	X
DTIC TAB	
Unannounced	
Justification	
By	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A	

A GENERALIZED

EXPERIMENT CONTROL AND DATA ACQUISITION SYSTEM.

THESIS,

AFIT/GCS/EE/81M-3

Arsenio R. Iloreta
Captain USAF

1 Mar 81

96

Approved for public release; distribution unlimited.

AFIT/GCS/EE/81M-3

A GENERALIZED EXPERIMENT CONTROL AND DATA ACQUISITION SYSTEM

THESIS

Presented to the Faculty of the School of Engineering

of the Air Force Institute of Technology

Air University

in Partial Fulfillment of the

Requirements for the Degree of

Master of Science

by

Arsenio R. Iloreta

Captain USAF

Graduate Computer Systems

March 1981

Approved for public release; distribution unlimited.

Preface

The Materials Laboratory of the Air Force Wright Aeronautical Laboratories (AFWAL/ML), Wright-Patterson Air Force Base, Ohio performs a myriad of experiments in which a great amount of data is being gathered. In some of these experiments, data gathering still involves manual interaction and/or the use of strip chart recordings or paper tapes. Digitizing the strip charts so that further manipulations of the data can be performed is laborious and prone to errors. To automate the data collection for a wide range of experiments, the Materials Laboratory purchased the necessary hardware components. It was about the time when all the components were received that I was assigned to the Computer Activities Group, which oversees the data automation needs of the Laboratory. I was assigned the project, and subsequently turned it into a thesis topic.

I wish to thank my supervisor and my Laboratory sponsor, Capt William H. Walker IV, for letting me, in fact encouraging me, to work on the project with a view towards writing it as a thesis. Acting as the critical user, he also provided helpful information when I needed it most. Special thanks goes to Colonel James P. Rutledge, my advisor and software engineering tutor, for his technical advice and direction; to the other members of the thesis committee, Major Alan Ross and Professor Thomas Hartrum, for their most welcome critique and comments; to Dane Hanby and Frank Beitel of the University of Dayton Research Institute, for providing assistance when seemingly trivial hardware and software matters escape me; and to A1C Chung Kong, for the "nuts and bolts" operation of the system. Most importantly, I wish to thank my wife,

Daisy; our children, Joan and Dennis; my in-laws, Jack and Lydia Yacas, without whose love, patience, understanding, support and encouragement, my efforts would surely have come up short and this project would not have come to fruition.

Contents

List of Figures

List of Tables

Abstract

A portable, automated, generalized experiment control and data acquisition system was developed for the Materials Laboratory to fulfill some of its data automation needs. The system is portable in that all hardware components fit in an aluminum carrying case, except for the terminal which is separate. The need for portability requires the program to be loaded from the terminal minicassette tape. The collected data is likewise dumped into the minicassette.

The hardware components were already procured before actual software development was begun, which included an extensive software requirements definition phase. The Air Force method of requirements definition, IDEF0, was used to model the software requirements. It involves a top-down approach to development. The model, thus diagrammed, provided the general structure from which the program itself was written. Structured code and ample documentation form the basic programming technique implemented. The program is currently operational, the system having been utilized in two actual experiment set-ups. The requirement to load the program from the minicassette was not satisfied, however, because of some loader error. The program can presently be loaded using a floppy disk system.

I. Introduction

Background

The Materials Laboratory of the Air Force Wright Aeronautical Laboratories (AFWAL/ML), Wright-Patterson Air Force Base, Ohio, performs a variety of experiments aimed at understanding the properties of materials. These experiments can be categorized into two basic types: (1) those that have dedicated, automated data acquisition systems designed (built) into them, and (2) those where data is gathered by manual observation and measurement or by using strip chart recorders. Both types of experiments have their limitations. Despite great strides in the reliability of automated data acquisition systems, these systems occasionally fail. Depending on the severity of the breakdown, it could take some time before the experiment can be continued or started again.

Data collection by manual observation and measurement is tedious and inherently inaccurate. Likewise, the use of strip charts can be as inaccurate if human interaction is required to read data off the charts (digitize) for further processing in order to determine the material property of interest. For instance, areas under strip chart curves are usually of high interest and must be manually determined.

The benefits of automating the data acquisition process for experiments requiring manual interaction are obvious (with due regard to the reliability problem mentioned above). This will not only free the experimenter to do other tasks but will also increase the speed and accuracy of the process. However, since experiments requiring manual

interaction are normally very highly specialized and are short-term or one-time in nature, the cost of procuring an automated system for each experiment is enormous. A portable, general, automated data acquisition system is therefore required. This general system can also serve as a back-up system to experiments of type (1).

The portable data acquisition system can additionally be used as an experiment controller, albeit a very primitive one indeed. By monitoring the values of a certain specified measurement, an appropriate predetermined action can be taken when these values get out of a given range. This action can be a message to the operator or it can be an automatic adjustment of an experiment parameter that the automated system controls.

It is not usually sufficient to proceed directly into designing a hardware and/or a software solution as soon as a need for an automated system is identified. Nor can the hardware and software efforts be totally removed from one another. Whereas hardware considerations may simply involve purchasing off-the-shelf equipments, software development involves more than simply designing the algorithm and then coding it in a suitable programming language. Too many software projects have encountered enormous cost and time overruns because the software requirements were never really defined. In fact, software costs have become a sizable portion of the Air Force's budget (Ref 10:12).

The Air Force, through numerous reviews and specification requirements, has begun to address the many phases of software development. These phases include system requirements definition,

software requirements definition, design, code and debug, test and integration, and operations and maintenance (Ref 2:1226-1227). Still, projects with varying degrees of difficulty meeting schedule and budget constraints are very frequently identified. Studies made of these projects have shown that they are lacking in the requirements definitions phase of the development (Ref 10). Requirements definition is the process of specifying what problem is to be solved. Obviously, without a clear understanding of what the system is to perform, the chances of something going wrong are increased tremendously. Incomplete or erroneous solution to a muddled requirement or correct solution to the wrong problem are the usual results.

Statement of the Problem

The purpose of this study is to develop the software required to implement the portable, automated, generalized experiment control and data acquisition system.

Constraints/Assumptions

Although the software to be developed must be applicable to most of the Laboratory's data acquisition needs, the Materials Laboratory has requested that the software be initially tailored for four of its projects, namely: (1) laser effect testing, (2) polymer cure kinetics, (3) liquid chromatography, and (4) mechanical testing. (See Chapter III, for further discussion on these projects.) This will limit the number of users from which input data to the software requirements process must be gathered. This is important because there must be

common agreement among parties on the set of specifications to be developed.

In addition, AFWAL/ML has already procured an LSI-11 microcomputer system and associated instrumentation to implement the hardware requirements. This constitutes an additional constraint because the software to be developed must be compatible with this microcomputer system.

Approach

The software requirements definition for the portable, automated generalized experiment control and data acquisition system are generated using the initial version of the Integrated Computer-Aided Manufacturing (ICAM) Definition Method (IDEF0), developed for the Air Force by SofTech, Inc. This technique, which is the Air Force version of the Structured Analysis and Design Techniques (SADT), involves modeling a system by means of a collection of diagrams showing things (objects or information) and activities (performed by men or machines). The model distinguishes what functions the system must perform from how the system is built to accomplish those functions (Ref 5).

The design of the software generally follows the composite/structured design techniques advanced by Myers in reference 9. Where deemed necessary, deviations are made and duly noted, giving specific reasons for such deviations. The code and debug, test and integration, and operations and maintenance phases of the development are in accordance with prevailing Department of Defense and/or Air Force

standards, including documentation standards, on a lot smaller scale (Ref 3).

Plan of Development

The requirements definition for the portable data acquisition system is presented in Chapter II. The software requirements definition with emphasis on the functional specifications is discussed in Chapter III. Chapter IV presents the formal functional specifications for the system. These specifications are in the form of activity and data models. Design of the software is the topic of Chapter V and the remaining development phases are included in Chapter VI . Chapter VII contains the summary and recommendations. A typical portion of the code, with emphasis on the documentation, a users manual and a printout of a sample session at the portable terminal are placed in the Appendix section.

II. System Requirements Definition

Introduction

Requirements definition encompasses all aspects of system development prior to actual system design, everything necessary to lay the groundwork for subsequent stages in system development. It deals with three subjects: 1) context analysis, which deals with the reasons why the system is to be created; 2) functional specification, which is a description of what the system is to be, in terms of the functions it must accomplish; and 3) design constraints, which includes a summary of conditions specifying how the required system is to be constructed and implemented (Ref 14:1-2). For this study, context analysis was discussed in Chapter I; functional specification is presented in a later section of this chapter, following the discussion of the design constraints.

Design Constraints

In order to take advantage of the Laboratory expertise on Digital Equipment Corporation (DEC) LSI-11 systems, the Materials Laboratory chose to base the portable, data acquisition system on the LSI-11/2 microprocessor. The other hardware components must, of course, be compatible with this microprocessor. A block diagram of the portable system is shown in Figure 1. The major component parts include, in addition to the LSI-11/2 microprocessor, a digital-to-analog and analog-to-digital converter (ADAC 1030), a multiplexer/amplifier board, a clock board, and the operator's terminal (Miniterm 1205). A complete

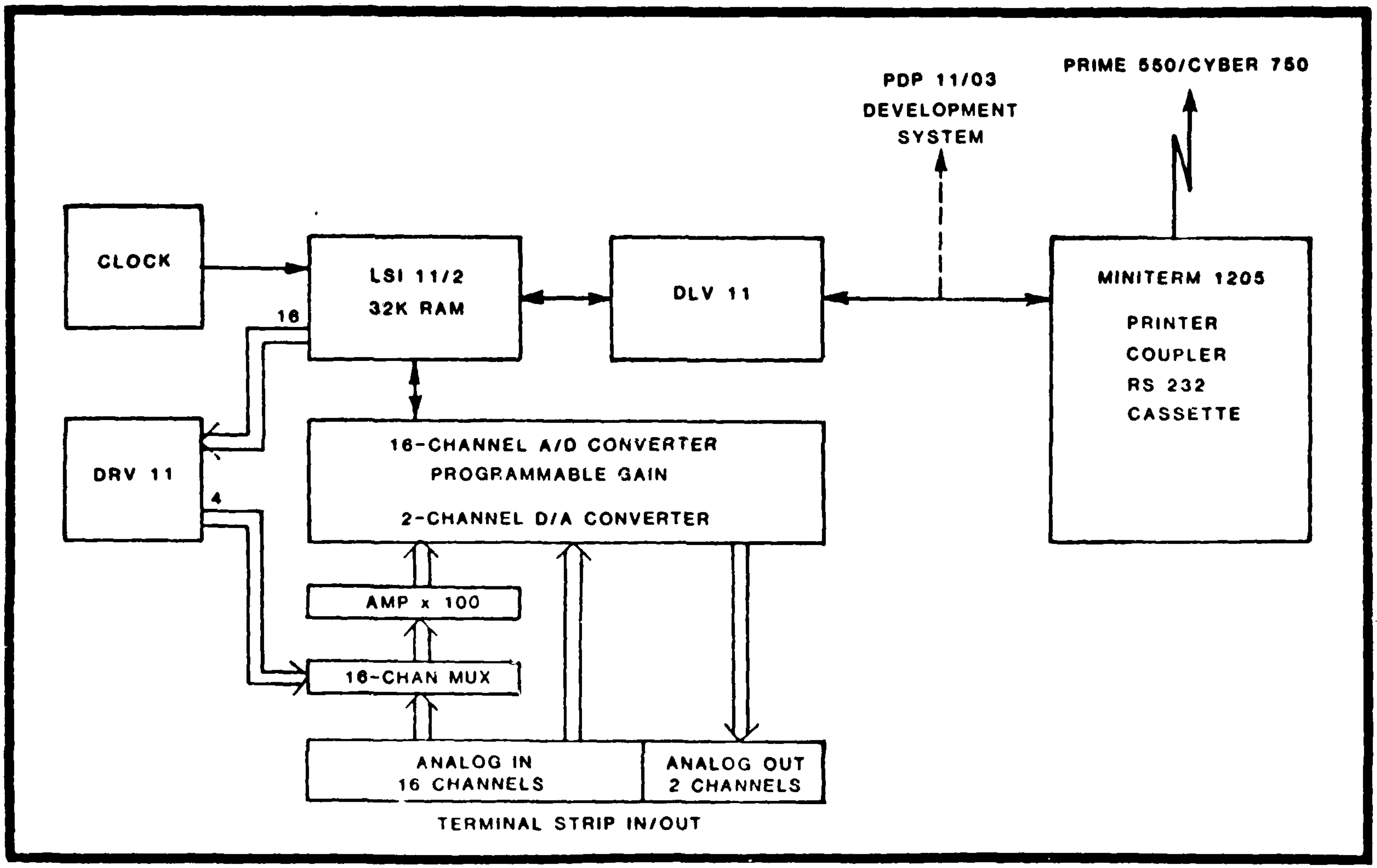


Figure 1. Portable Data Acquisition System

parts listing for the system is given in Table I.

Table I. Portable Data Acquisition System Components

Part/Description	Manufacturer	Part Number
LSI-11/2 Central Processing Unit	DEC	KD11-HA
32K RAM Memory	DEC	MSV11-DD
Backplane/Card Guide Assembly	MDB Systems	MLSI-BPA84
Power Supply	Standard Power Inc.	SPS 130D 5/12
A/D and D/A Converter	ADAC Corporation	ADAC 1030
Parallel I/O Port	DEC	DRV11
Serial I/O Port	DEC	DLV11
Floating Point Chip	DEC	KEV11
Portable Terminal	Computer Devices	Miniterm 1205
Multiplexer	Harris	HI3-0506-5
Instrumentation Amplifier	Analog Devices	605K-100
DC/DC Converter	Analog Devices	940
Carrying Case	Zero Corporation	110-P6
Clock Board		

LSI-11/2. The central processor for the portable data acquisition system is DEC's LSI-11/2 (part number KD11-HA). It is a microprocessor with reasonably fast speed and has 32K words (16 bits) of memory. It supports the basic PDP-11 instruction set and, with the optional Floating Instruction Set (KEV11) (which was also purchased), can perform floating point arithmetic operations.

ADAC 1030. The ADAC 1030 is a 16-channel analog-to-digital plus an additional two-channel digital-to-analog converter. It is configured to have fully differential analog inputs with a range of -10 volts to +10 volts. This configuration is jumper selectable and was chosen to accommodate the more common range of analog output signals found in laboratory devices. The weaker signals (millivolt range) are amplified before they are fed to the converter. Likewise, the selected output range of the digital-to-analog converter is -10 volts to +10 volts. Selection is also through the use of jumper wires.

Multiplexer/Amplifier. The multiplexer/amplifier is comprised of a multiplexer, an instrumentation amplifier, and a DC/DC converter. The particular circuit that is used in this project was designed and built by Dane Hanby of the University of Dayton Research Institute and is shown in Figure 2. The output of this circuit is fed directly to the ADAC A/D converter, using the input pins for channel 1 as shown in Figure 3.

Clock Board. The clock board provides the timing interrupt circuitry for the system. It consists of a crystal oscillator with TTL output,

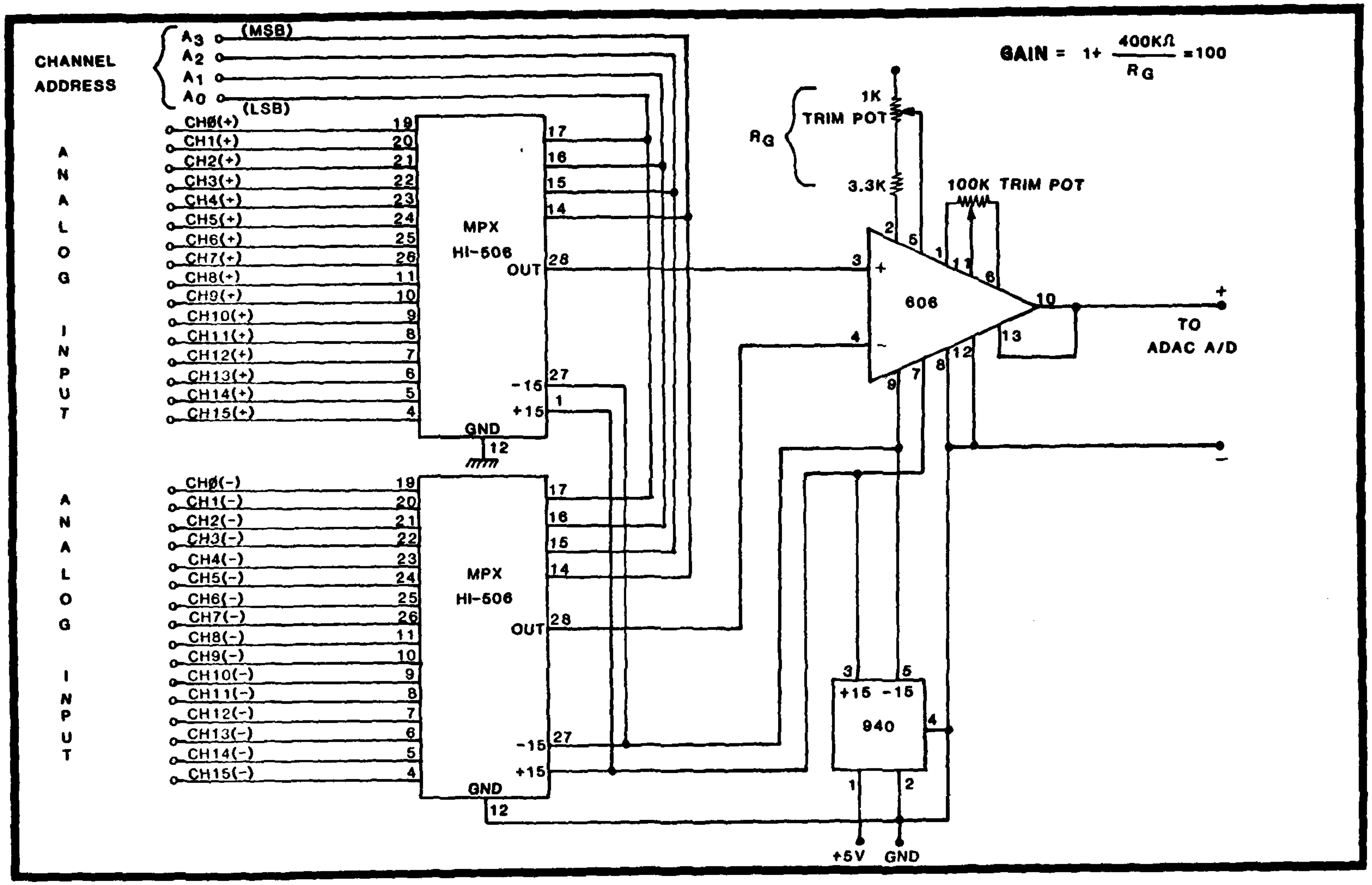


Figure 2. Multiplexer/Amplifier Circuit Diagram

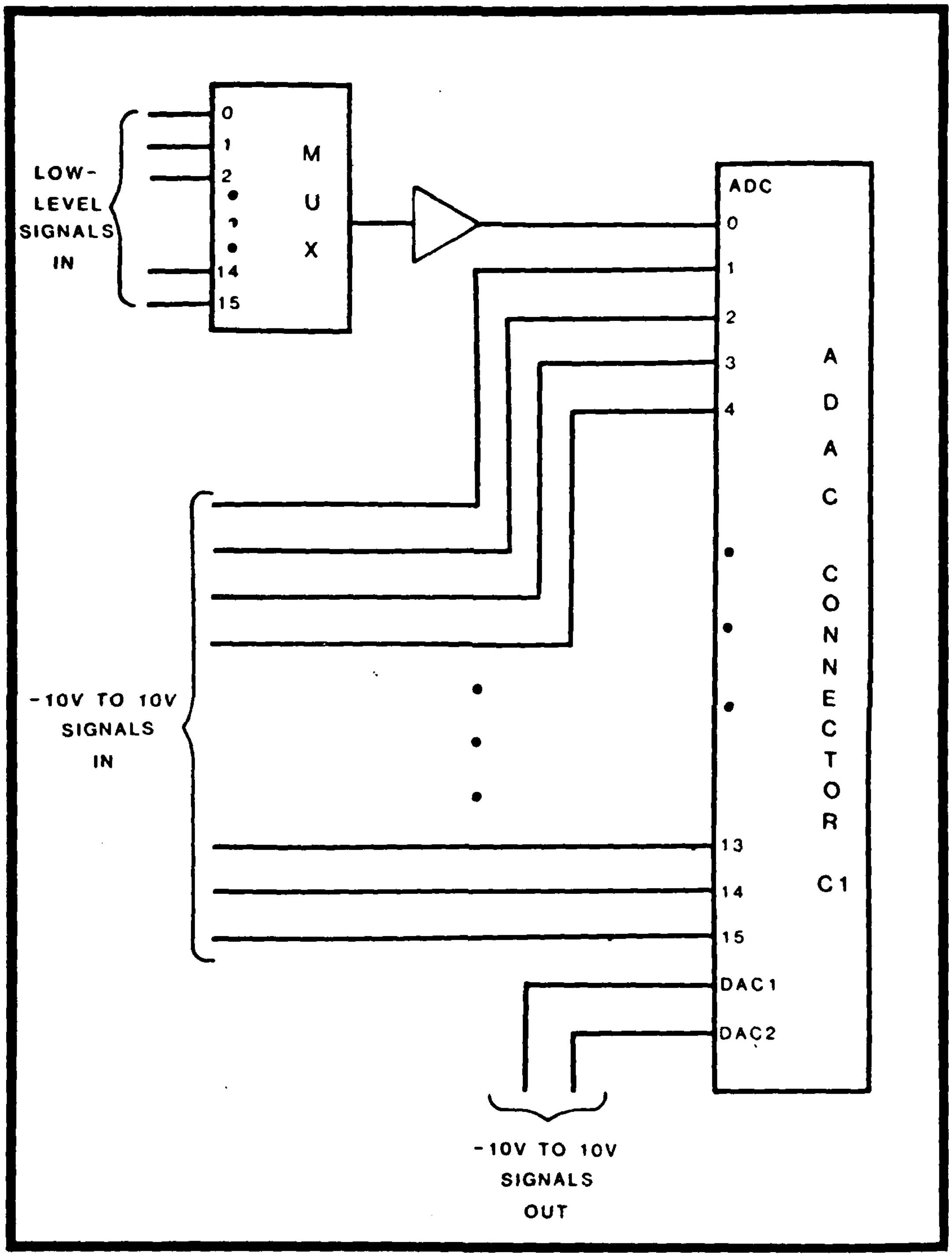


Figure 3. A/D and D/A Pin Connections

three synchronous decade counters, and a line driver. The diagram (Figure 4) shows the interconnections among the circuit parts as well as the interface to the system. This circuit was designed by Tom Wood while he was working for the University of Dayton Research Institute.

<u>Miniterm 1205</u>. This is the portable terminal that allows operator communication with the microcomputer system. It provides for Standard RS232 interfacing, has a built-in acoustic coupler capable of 300 or 1200 baud transmission rates, and includes 8K of RAM memory (expandable to 32K) making it possible to edit data off-line perhaps after or in between experiments. It also includes a minicassette tape drive to provide extra storage of special programs or historical experiment data of up to 68,000 characters per minicassette.

The entire hardware of the system is contained in a single carrying case. The operator's portable terminal is separate and is self-contained. Set-up requires:

a) connecting power cord to 110 VAC;

b) connecting cable between the portable terminal and the hardware case;

c) wiring analog in/analog out experiment lines to proper terminals on terminal strip of hardware case;

d) for connection to the remote host computer, establishing communication line and placing telephone receiver in acoustic coupler receptacle.

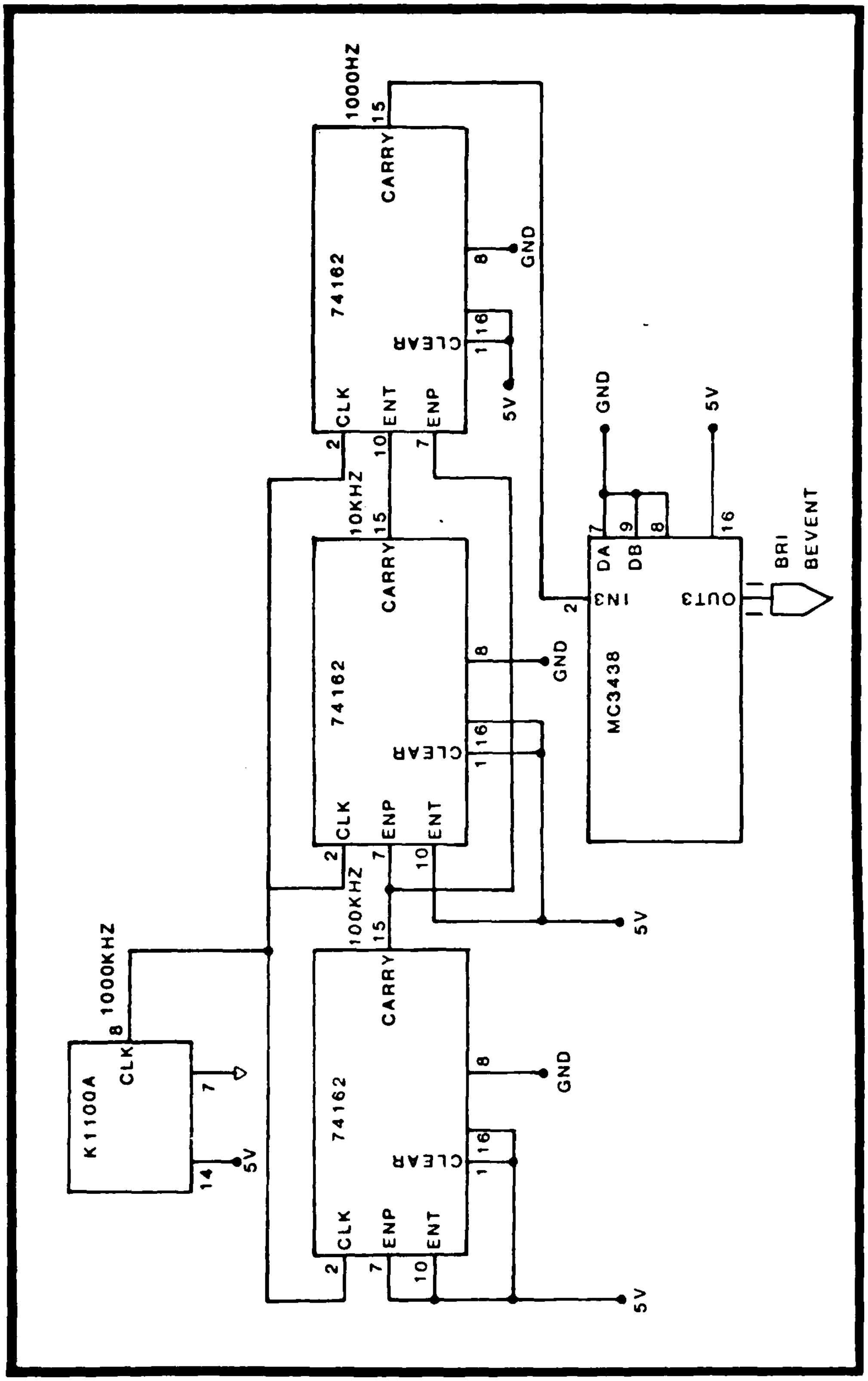


Figure 4. Timing Interrupt Circuit

Functional Specifications

As mentioned in Chapter I, the Materials Laboratory's objective is to assemble a general, portable data acquisition and experiment control system to support mission essential part-time and short-term projects. This system is to operate in two basic modes: 1) data acquisition/process control mode, and 2) data communication mode.

In the data acquisition/process control mode, the system is wired to the experiment and the proper data acquisition and control program is loaded from the operator terminal's cassette tape. Program parameters are set by the user from the terminal keyboard. Control of the experiment is initiated and data acquisition is started. After data collection is completed, data is transferred from system memory to cassette tape and/or to the appropriate host computer as explained in the data communication mode.

In the data communication mode, the user transmits data stored in memory or cassette tape from the Miniterm to the computer which will perform data reduction and/or provide long-term storage. This mode is also used to transfer the absolute code for programs from the PDP-11/03 development system to the portable system (cassette tape) or programs from cassette tape to memory.

Software development was performed on the PDP-11/03 development system and transferred to cassette tape for portability. Note that initial program checkout may be performed at the development system site, with all the system interfaces actually available.

Summary

This chapter covered the functional specification and design constraint aspects of system requirements definition. The context analysis, it was mentioned, has already been discussed in Chapter I. The rest of the thesis will now focus on the software development for the general, portable data acquisition system.

III. Software Requirements Definition

Introduction

Neglect of the requirements definition phase has been blamed, along with other causes, for excessive cost and schedule overruns. In addition, the elimination of this important step often results in an incomplete documentation package which, in turn, contributes significantly to operations/maintenance problems later in the software life-cycle.

The software for this thesis was generated to implement the portable data acquisition system, the hardware component parts of which have already been procured by the sponsoring laboratory. The procurement of the hardware components represents a software design constraint inasmuch as the software product must be compatible with the procured hardware system. Also, while there are numerous packaged routines available for such systems (LSI-11 systems), the portability requirement represents an additional design constraint, because not all these routines are usable for stand-alone systems. This chapter describes the context analysis and the functional specification aspects of the software requirements definition.

Context Analysis

The software requirement is for the implementation of the portable data acquisition system. Four Materials Laboratory projects were initially identified as possible applications for this system. These

are 1) laser effect testing, 2) polymer cure kinetics, 3) liquid chromatography, and 4) mechanical testing. In order to develop a set of software requirements, we must examine the individual requirements of these four example projects.

Laser Effect Testing. This experiment determines the thermal effects of firing a laser beam at a given material specimen. Several thermocouple beads are positioned in a sample to record the temperature profile of the material as it is burned. Presently, the signals from the thermocouple wires are recorded on a VISICORDER chart. In order to perform mathematical manipulations of the collected data, the charts must be digitized. At most ten signals are monitored for each experiment run. The run itself takes approximately ten seconds, with the laser beam normally on for only 0.3 second.

There is a short period after the run is started that no meaningful data is available for collection. The start of the data collection process thus requires the detection of a START signal, which is not presently available. One hundred data points per thermocouple wire meet the digitizing requirement; which means approximately one thousand data points must be recorded every ten seconds.

Polymer Cure Kinetics. The Polymer Branch performs polymer cure kinetics using thermal analysis instrumentation. A polymer sample is heated at a given rate and the temperature of the cell in which it is placed is recorded at specific time intervals, usually on the order of one second. Presently, this temperature data is recorded on paper tape and, for back-up and further comparison, on a strip chart. For further

data manipulations, the paper tape is read into the host computer where the necessary application programs are installed. The data is plotted; the area under the curve is determined; and after a series of curves, a time-temperature-degree-of-cure relationship is established for the polymer.

Five different heating rates are used in this experiment (80, 40, 20, 10, and 5 degrees Kelvin per minute) to get to a cell temperature of 400 degrees Kelvin. Between 350 and 500 temperature readings are collected per scan which means that, for the shortest experiment scan (five minutes), at most one hundred of these readings are recorded per minute.

Liquid Chromatography. High performance liquid chromatography is used in the Materials Laboratory for research and routine quantitative analyses. A strip chart recording is the output medium used for collected data. Reading the charts and manually determining the area under each "peak" in the curve (extrapolated accordingly if peaks overlap) is the standard manner in which the relative amount of each sample's components is measured. A normal run takes approximately two minutes and 200 data points render satisfactory results.

Mechanical Testing. The Metals Behavior Branch performs experiments to study the effect of cracks on the strength of materials. Samples are placed in a loading frame in which a variable load can be exerted on the sample. The samples are notched or cut perpendicular to the axis of loading to promote cracking. The rate of crack propagation is determined in part by the amplitude of the load being applied. These

experiments sometimes take an inordinate amount of time because the cracks do not propagate very fast; or, conversely, end abruptly because the samples fracture easily with the applied load.

A typical crack growth experiment usually takes between eight hours to almost three weeks. Twenty blocks of crack growth data, corresponding to twenty different temperatures, are gathered. Each block contains 50 to 150 data points. These data points are a function of the specimen geometry, the applied load, crack length, and time. The relationship between these factors is predetermined and must be maintained through the experiment run. Since the specimen geometry remains essentially constant, the applied load must be "controlled" according to the rate of crack propagation (change in crack length over time). This control requirement is normally performed at one- to ten-minute intervals.

Because of the time element involved and the inherent inaccuracies of the manual process, the need is clear for an automated system. Automating each experiment's data collection process according to its individual requirements is economically unfeasible. Therefore, any automated system to be developed must be general enough to accommodate at least the range of experiments discussed above. It must also be portable if it is to be easily moved from one experiment location to another.

Projects similar to those mentioned above abound in the Laboratory and can be easily adapted to utilize a portable, automated data acquisition system. In addition, the dynamic nature of the Laboratory's

commitment to materials research makes it very desirable to build such a general system.

Functional Specifications

Any system developed must be a general-application system. It must provide the user the opportunity to define his particular experiment's data collection and experiment control needs. The capability to record data at any given time is needed in the system. In addition, there is a need, as in the laser experiment, to provide a mechanism to start data collection separately from the start of the experiment; that is, to take action depending upon a given condition. The mechanical testing experiment exhibits another sort of need: the ability to provide feedback to the experiment to control a particular parameter. These needs make up the functional specifications discussed below.

Define Experiment. When the system is first put into operation, a menu of all the valid commands, including their descriptions, is shown to the user. Among these is a DEFINE command that allows the user to define the data channels he will use in collecting data from his experiment. This command is appropriately labelled as the first command given in normal system operation. Up to three channel files can be defined: channel schedule file, channel dependency file, and channel control file. A channel schedule file must always be defined; the other two are defined only as required.

The channel schedule file consists, as a minimum, of the information required to collect data at some indicated times. This

information may include the initial read time and the time interval between reads for each channel defined. The channel dependency file, on the other hand, is defined for experiments where certain information does not become meaningful until some condition arises; for example, a voltage reading in an experiment may be meaningless unless the temperature reading in channel 1, say, is over 350 degrees Farenheit when certain chemical reactions are known to take place. The dependency file requires the channel (number) that must be monitored for the condition, the condition itself, and the channel that must be read when the condition is satisfied.

A channel control file is needed when the system is to be also used as an experiment controller. In this configuration, one or more of the scheduled channels (defined in the channel schedule file) may be monitored and their values compared against a given range. One of two actions is taken when this value does get out of range: a message is printed on the operator's terminal or a specified output channel is adjusted accordingly. The control channel file therefore includes the low and high values of the required range, the appropriate action when the control channel value gets out of this range, the output channels when the control channel value is below and over the range, respectively, and the message text, if operator interaction is required.

For each of the above channel definitions, the user is given the opportunity to check his entry lines and modify them as he wishes. These change options include the ability to change any single line, change all lines, add a new line, or delete an existing line.

Set/Change Parameters. When the portable automated data acquisition system is also to be used as an experiment controller, a function to set, reset, or change experiment parameters is required. This mechanism can either be manual or automatic. The control channels are monitored and their values continually compared against given ranges. When a channel reading does get out of its specified range, the proper adjustments must be made.

Acquire Data. Within its speed capabilities, the system must be able to sample the analog signals present on the defined data channels and convert these into digital data. These read times are available from the channel schedule file and readings must be made as closely as possible to the indicated read times. Once a scheduled channel is read, its next read time is updated using the time interval between channel reads. If there is enough time before the next scheduled read time, the channel dependency file is also checked. Any dependent channel that has its dependency conditions satisfied must be read, and its read time subsequently updated. If a control channel file is present, it, too, must be checked.

Store Data. If, after reading a scheduled or a dependent channel, its internal storage capacity is reached, the system must be able to dump whatever data has been collected to some permanent storage medium. Data collection can then proceed as before.

Depending on the time interval between scheduled channel reads, some data may be lost while storing one batch of data into the

terminal's recording equipment. The operator is given the option to transmit the acquired data before the system's internal storage (memory) is full, so that shorter, but obviously more, time intervals will be spent transferring his blocks of data.

Summary

This chapter discussed the context analysis step in the software requirements definition phase. It also presented the software functional specifications in a general, informal manner. Software functions included defining the experiment, controlling parameters, acquiring data, and storing data. These specifications will now be formally presented in the next chapter.

IV. Functional Specifications

Introduction

The formal functional specifications for the system are developed using the initial version of ICAM Definition Method (IDEF0). Its fundamental concept is a combination of structured analysis concepts. Two of these concepts are to understand a system by creating a model that graphically shows things (objects or information) and activities (performed by men or machines) and to structure a model as a heirarchy with major functions at the top and successive levels revealing well-bounded details. In IDEF0, this model is in the form of diagrams which are composed simply of boxes and arrows, boxes representing activities and arrows representing things (Ref 5).

Creating the diagrams for the proposed data acquisition system is facilitated by the discussion of the major functional requirements in the previous chapter. This chapter presents the subdivision of these functions into smaller subfunctions, in the form of diagrams, which make up the model for the system.

Activity Model

The initial step in the discussion of the activity model is to present the model in the form of a node index. Here the relatonships among the different nodes comprising the model is exposed, and understanding the model is enhanced. This node index, showing the order

of the node diagrams, is shown in Table II. The detailed functional requirements are given in the following diagrams, starting with node A-0, Manage Data Acquisition.

Table II. Node Index

- A0 Manage Data Acquisition
  - A1 Define Experiment
    - A11 Determine File Type
    - A12 Create Schedule File
    - A13 Create Dependency File
    - A14 Create Control File
  - A2 Set/Change Parameters
    - A21 Check Control Value
    - A22 Determine Correct Setting
    - A23 Automatically Adjust Channel
    - A24 Manually Set Parameter
  - A3 Acquire Data
    - A31 Read Scheduled Channel
    - A32 Update Schedule File
    - A33 Read Dependent Channel
    - A34 Update Dependency File
  - A4 Store Data
    - A41 Generate Start-Receive Signal
    - A42 Receive Data
    - A43 Generate Stop-Receive Signal

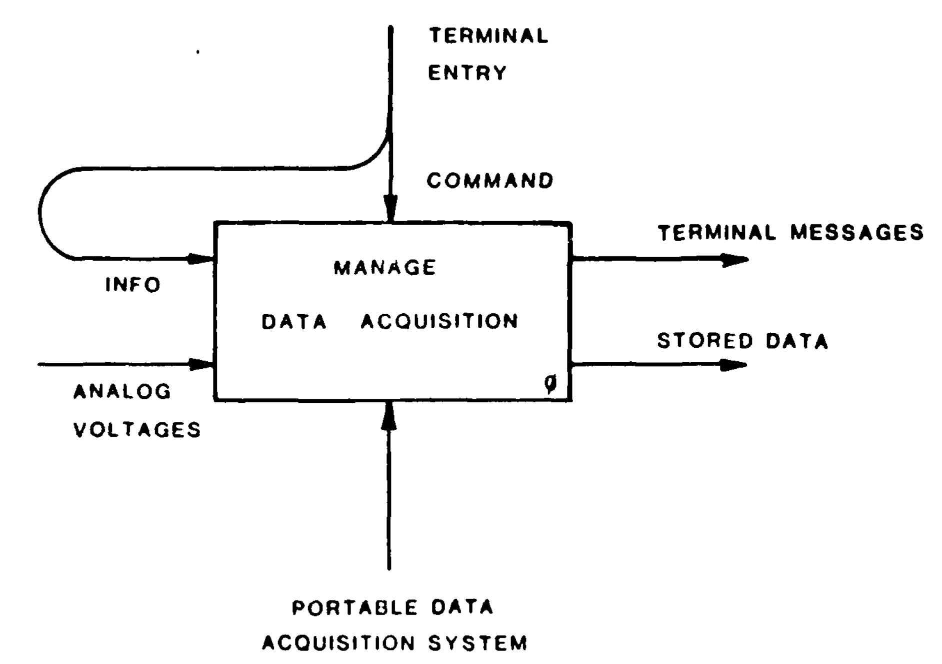


Figure 5. Node A-Ø, Manage Data Acquisition

The portable data acquisition system is able to conduct the experiment and collect data through commands given by the operator. Inputs include operator information provided via his terminal and the analog voltages supplied from the experiment output lines. Messages may be given to the operator for proper (corrective) actions, while the collected data are written onto a minicassette tape available on the operator's portable terminal.

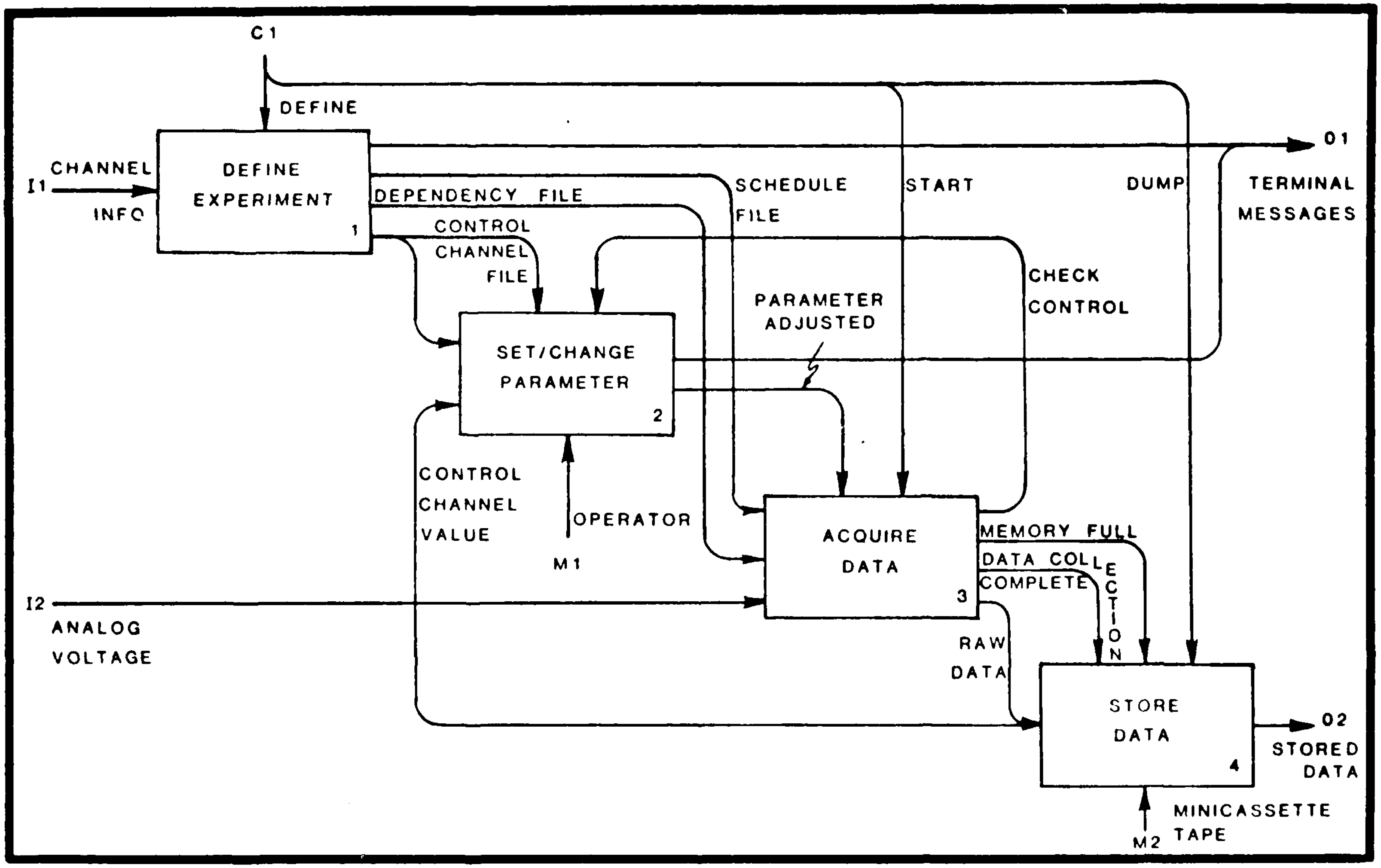


Figure 6. Node A$\emptyset$, Manage Data Acquisition

The acquisition of data and the control of the experiment consists of four major functions: defining the experiment in terms of the sampling strategy for the selected data channels (1); setting the necessary equipment parameters to the desired values, whether manually or automatically (2); actually reading the data within the specified time frame (3); and storing the data for transfer to the mainframe computers for archiving and/or further reduction (4). Define Experiment provides channel information to both the Set/Change Parameter and the Acquire Data functions: to make sure that the channel parameters are within the correct bounds and to provide the proper time frame for reading the channels, respectively. Acquire Data is also dependent upon the generation of the proper signal in Set/Change Parameter. When data gathering is complete, when system memory is full, or when directed by the operator, the data collected is stored by dumping it to the portable terminal's minicassette.

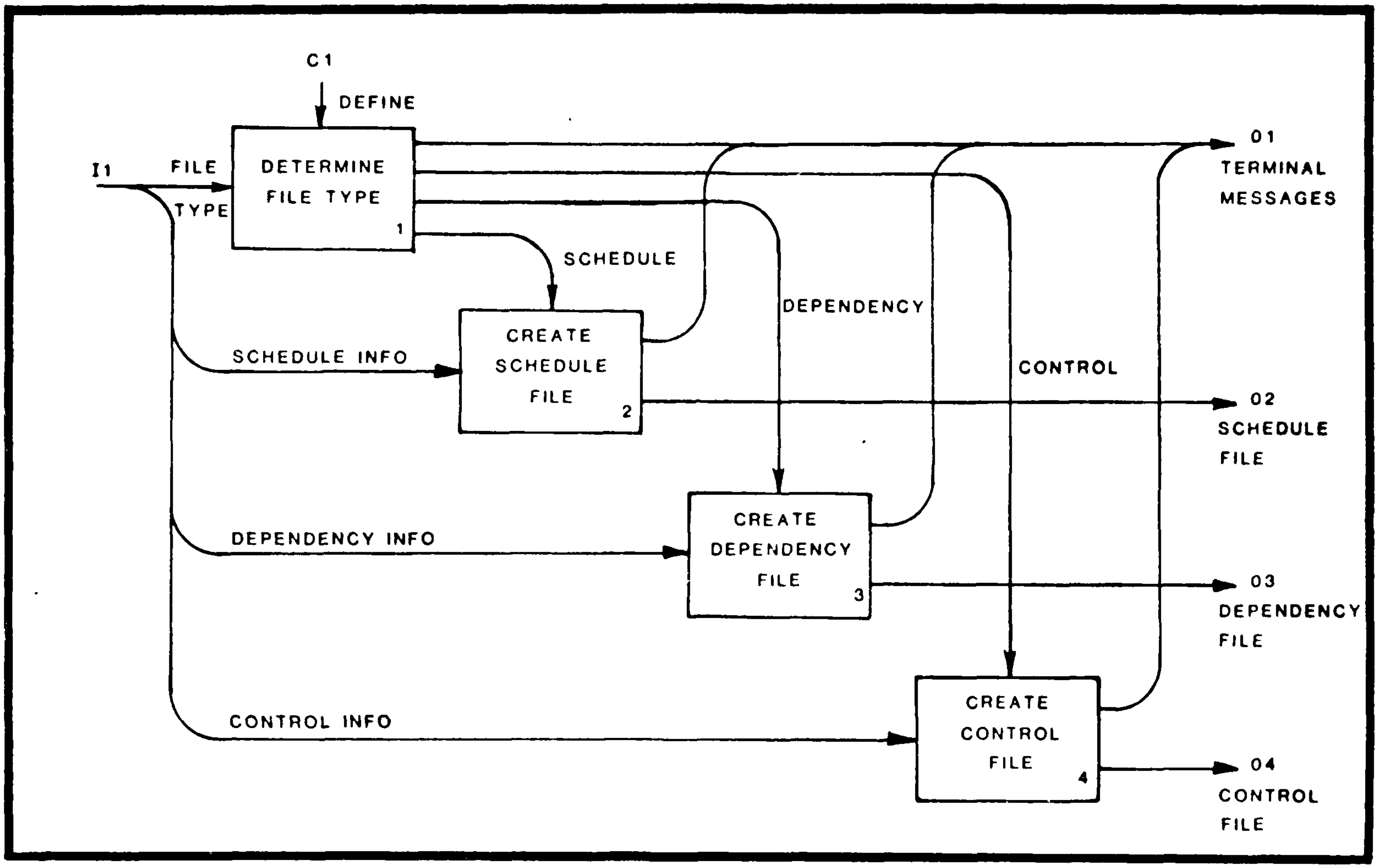


Figure 7. Node A1, Define Experiment

Define Experiment provides the necessary sampling strategy and control information in terms of three channel files. Three subordinate functions create these files: Create Schedule File (2), Create Dependency File (3), and Create Control File (4). While the contents of these files are different, the manner in which they are created is not. Once a file type is determined (1), the file is built up by having the operator type in a line of information according to the format shown in his terminal.

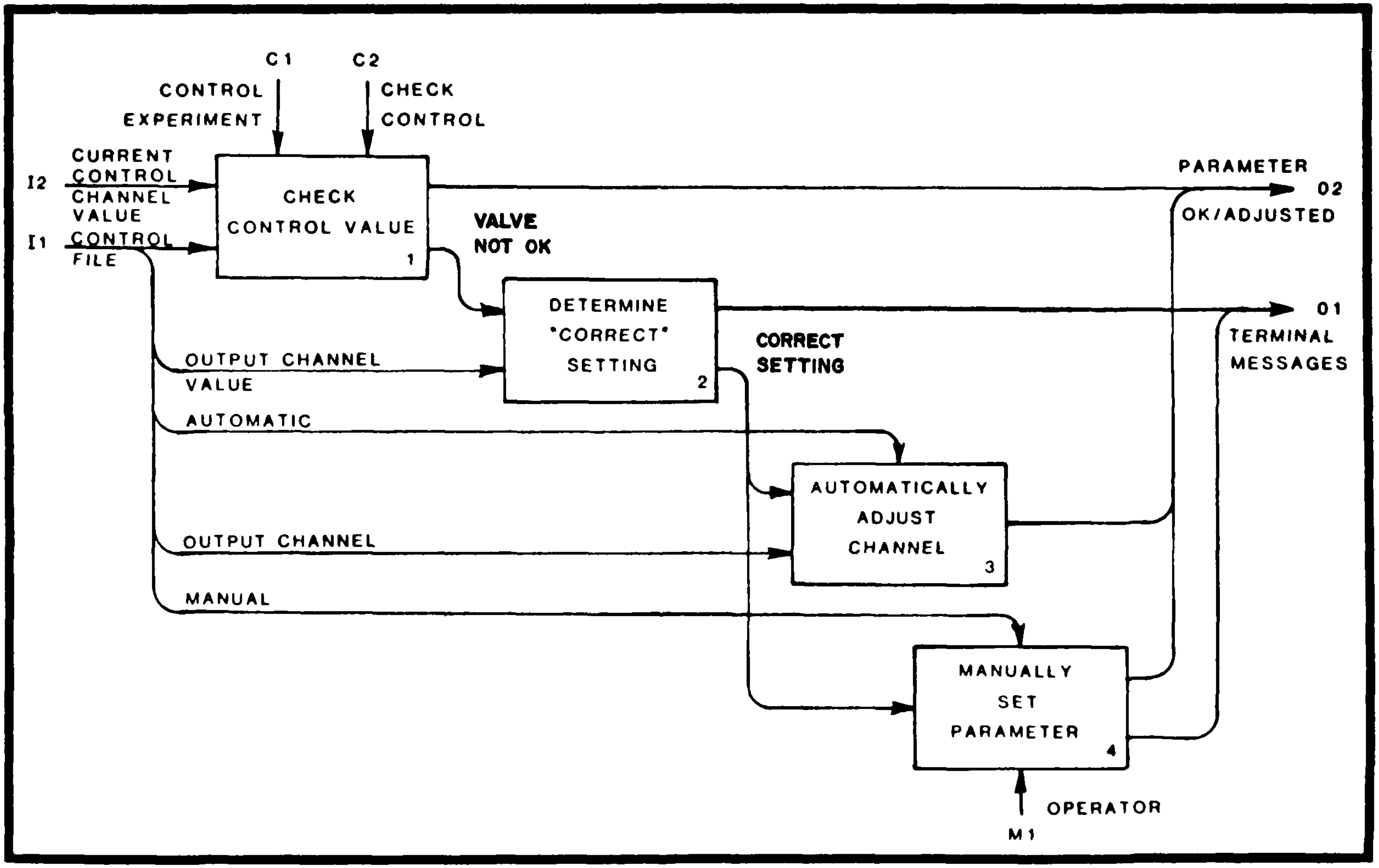


Figure 8. Node A2, Set/Change Parameter

One of the files generated by the Define Experiment function is a channel control file. This file is checked every time a control channel is read (1). This value may or may not be within the correct range for proper controlling of the experiment (2). If not, a parameter associated with this channel must be properly adjusted, either automatically (3) or manually (4).

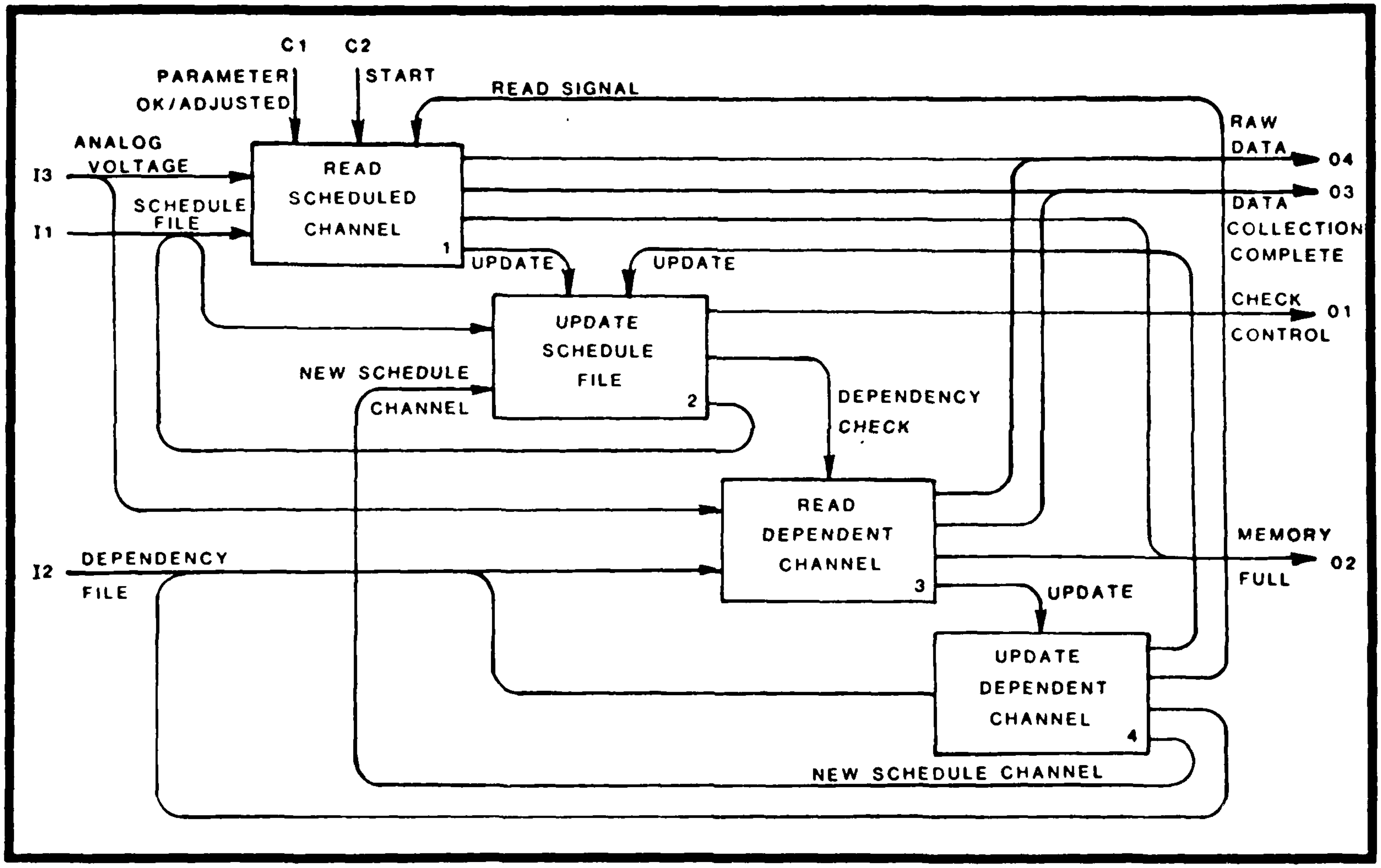


Figure 9. Node A3, Acquire Data

Data is acquired by reading the scheduled channels during the appropriate times (1). Other channels may also have to be read during this time frame, if certain conditions are met. These are called the dependent channels (3). The schedule file and the dependency file are updated (2 and 4) during this activity to reflect the next scheduled time when these channels are to be read again.

The analog signals are experiment output signals that must be monitored. When these signals are weak such that they are not capable of driving the analog-to-digital converter, they are first amplified to the proper range. This is a hardware function and is not shown in the diagram.

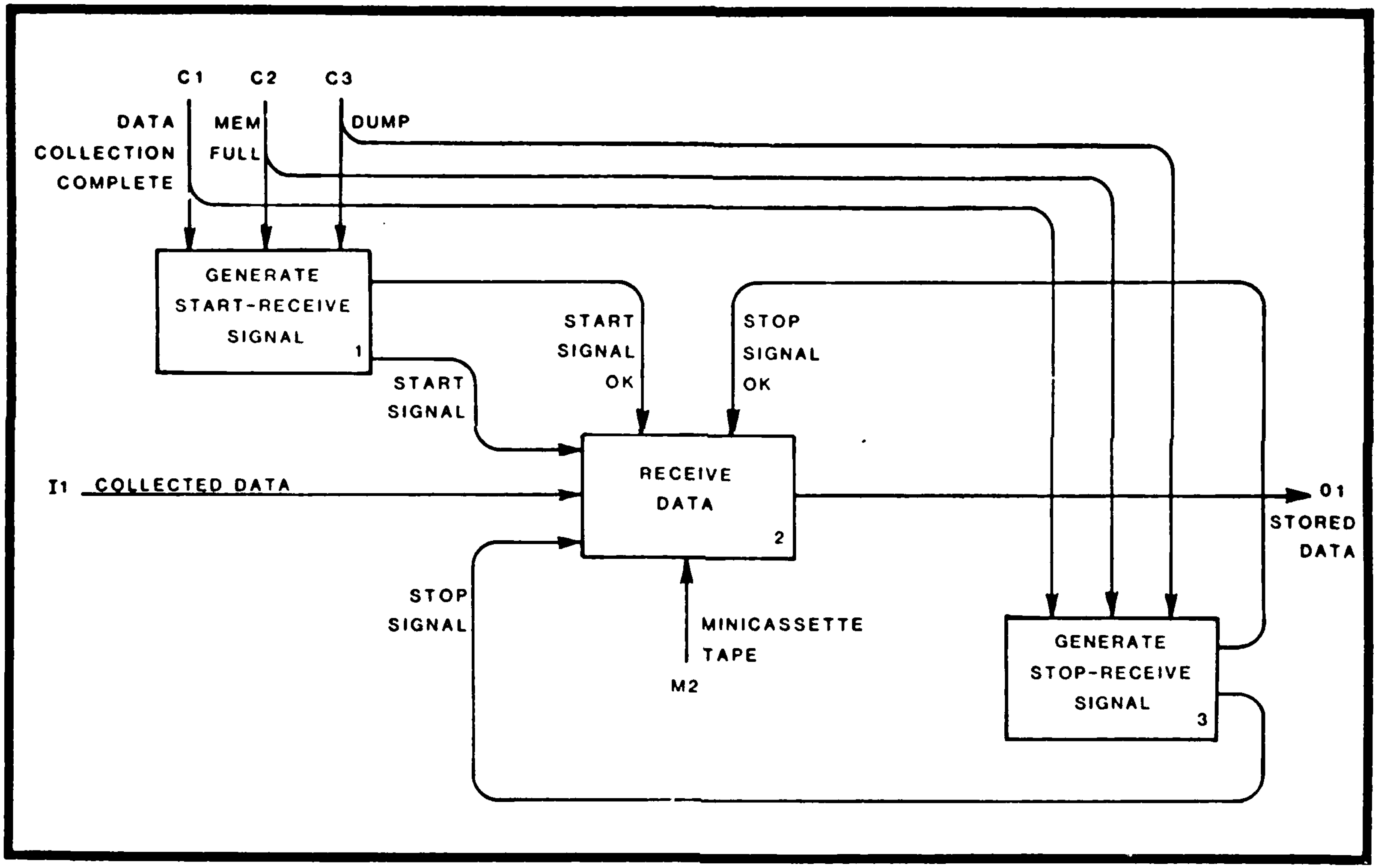


Figure 10. Node A4, Store Data

Once data acquisition is completed, or when the system memory is full, or when directed by the operator, the data is dumped from the system memory to the minicassette available on the operator's terminal. This requires a Start-Receive signal (1) to activate the tape drive mechanism for the proper receipt of data (2). A Stop-Receive signal (3) is generated when all of the data is received. Care must be taken in order not to overflow the capacity of the minicassette. As much packing of integer data as possible must be incorporated into the program for maximum use of the minicassette, especially when a lot of data has to be taken.

Summary

This chapter presented the formal functional specifications for the portable data acquisition system in terms of the diagramming conventions of IDEF0. The relationships among the different nodes and the flow of data between them are now evident. These diagrams provide a very good foundation towards designing the software. A set of specifications has been developed against which the software can be compared.

V. System Design

Introduction

A principle of structured analysis states that an exposition of a certain problem can be performed recursively through a systematic breaking down of the subject matter into six or less component parts (Ref 5:Chapter 2,2-2). It is a gradual, top-down decomposition approach, leaving nothing out at any stage. The end result is an analysis expressed in well-structured diagrams that enhance an understanding of the requirements for the solution of the original, complex problem. These requirements can then be translated, in a rigorous, organized, efficient and, above all, understandable fashion, into actual system design.

The diagrams of the previous chapter lend themselves well to the design of the software for the portable data acquisition system. The four major functions are grouped into "run preparation" and "run experiment" categories according to their time of occurrence. Define Experiment falls into the first category, as well as the initial setting of experiment parameters. Data collection and any follow-on setting/changing of parameters occur during the actual running of the experiment. Data storage (dumping into tape) may be done while the experiment is running; usually, however, it occurs after the experiment has terminated.

Using the same heirarchical approach as IDEF0 proposes, the basic functions are designed and the results presented in this chapter.

Structure Charts

The overall structure of the software system is shown in Figure 11. Each box represents a (program) module; that is, a collection of executable program statements which is a closed subroutine, can be called from any other module in the program, and has the potential of being independently compiled (Ref 9:11). Note the striking similarity between the program structure and the functional specification diagrams of the previous chapter. This is no coincidence. The ultimate objective of structured analysis and design is to eliminate the ambiguities and the oftentimes random relationships between the modules that make up the whole program. In the structure of Figure 11, each module has its own particular function to perform, which is nicely broken down into simpler component functions by lower-level modules (Figures 12-26). The lowest-level modules perform elementary functions.

The data flow among the program modules is exhibited in tabular form under each decomposition figure in the manner suggested by Myers (Ref 9:110-121). The IN and OUT columns refer to input and output data, respectively; the numbers refer to the interface labels shown. An input datum is something whose value is significant upon entry to the called module whereas an output datum is something whose value is assigned or changed within the called module. It is therefore possible for a data item to be both an input and an output (Ref 9:15).

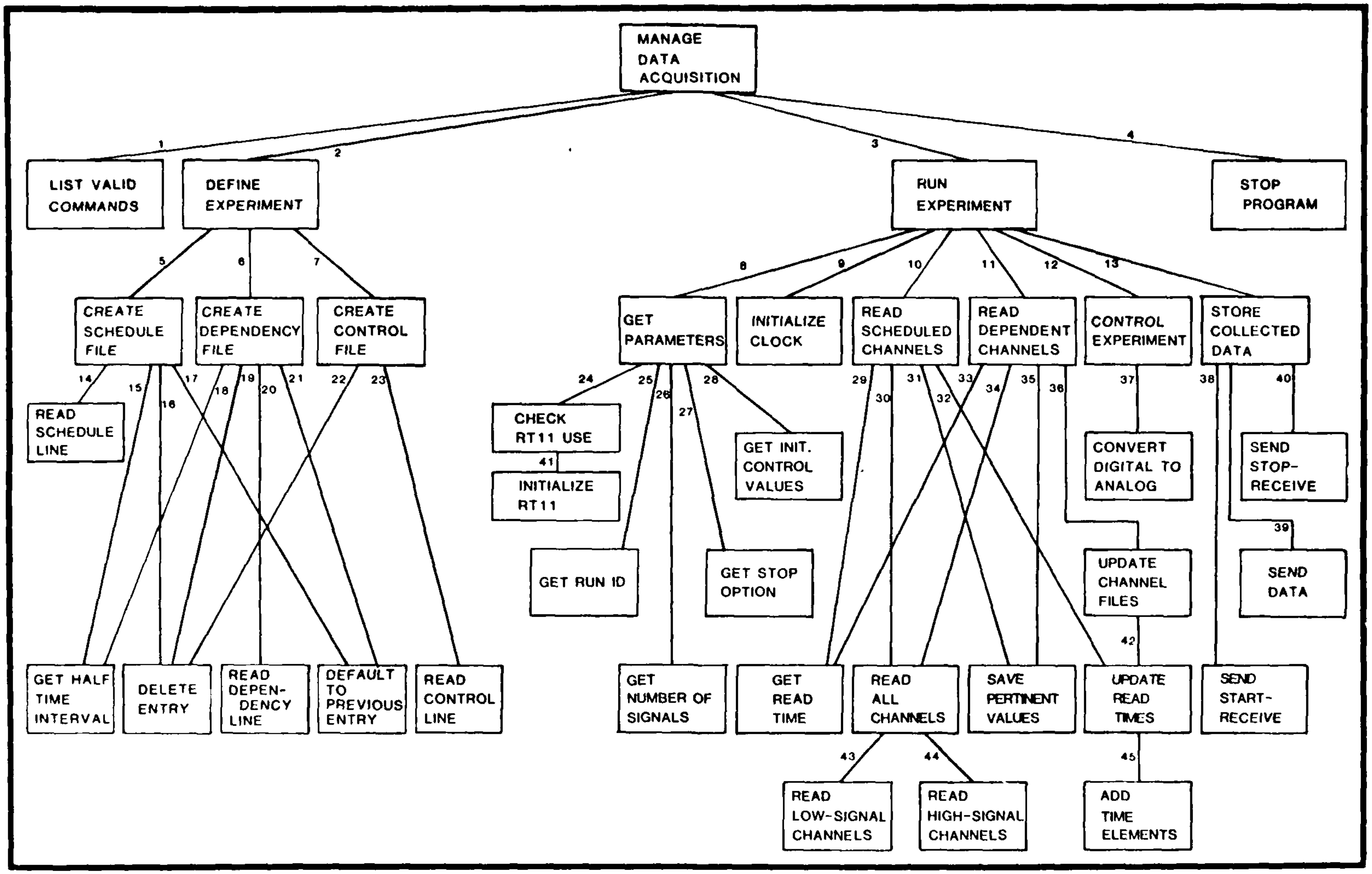
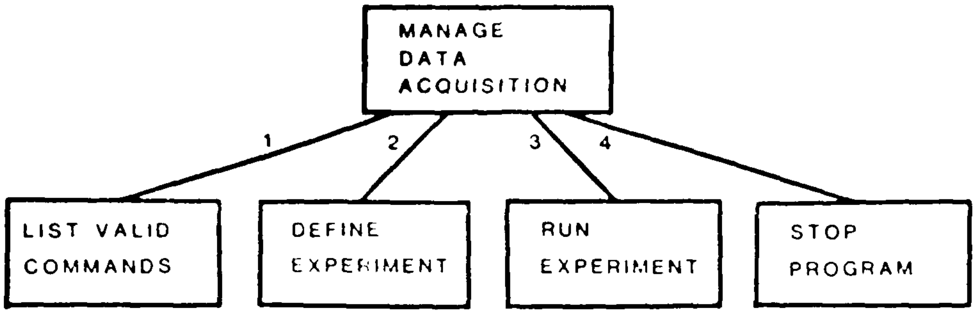
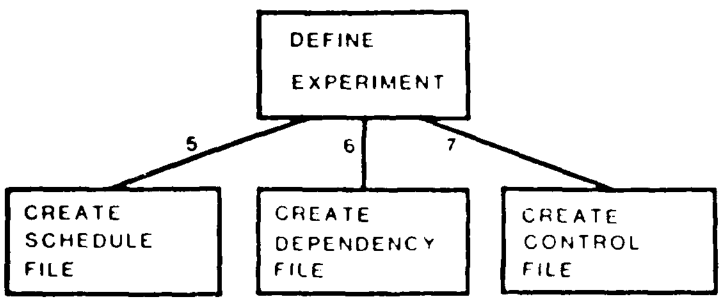


Figure 11. Program Structure for the Portable Data Acquisition System



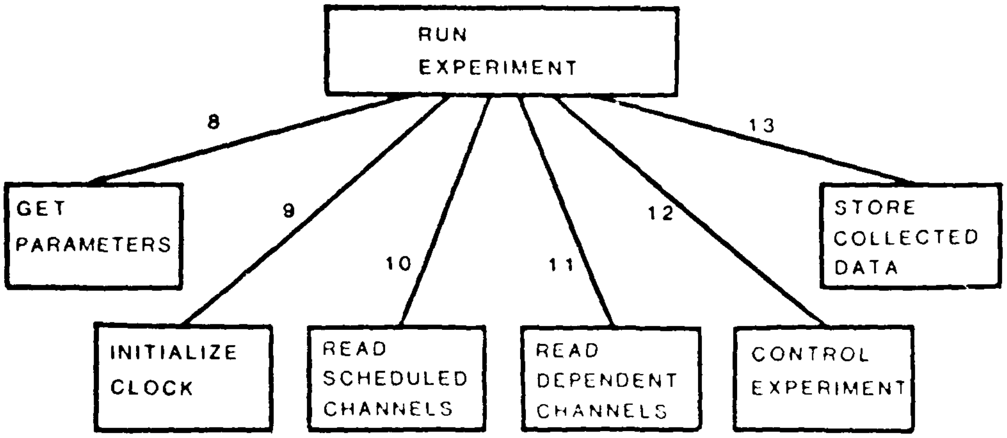
	IN	OUT
1	VALID COMMAND INDICATOR	VALID COMMAND INDICATOR
2	VALID COMMAND INDICATOR	VALID COMMAND INDICATOR CHANNEL SCHEDULE FILE CHANNEL DEPENDENCY FILE CHANNEL CONTROL FILE
3	VALID COMMAND INDICATOR CHANNEL SCHEDULE FILE CHANNEL DEPENDENCY FILE CHANNEL CONTROL FILE	VALID COMMAND INDICATOR
4	(NOT A PROGRAM MODULE)	(NOT A PROGRAM MODULE)

Figure 12. Decomposition of "Manage Data Acquistion".



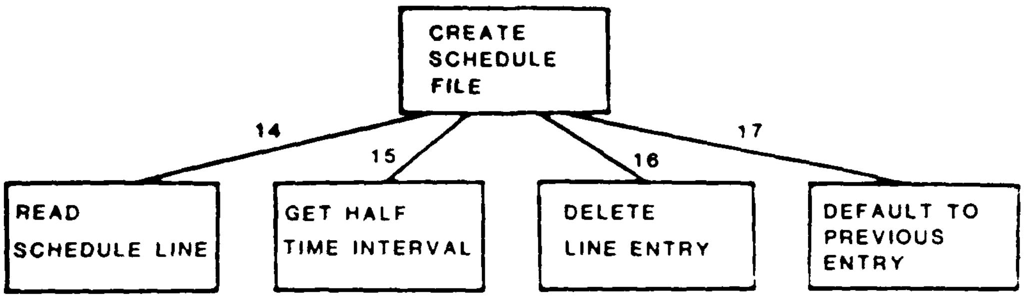
	IN	OUT
5	- - -	CHANNEL SCHEDULE FILE
6	- - -	CHANNEL DEPENDENCY FILE
7	- - -	CHANNEL CONTROL FILE

Figure 13. Decomposition of "Define Experiment"



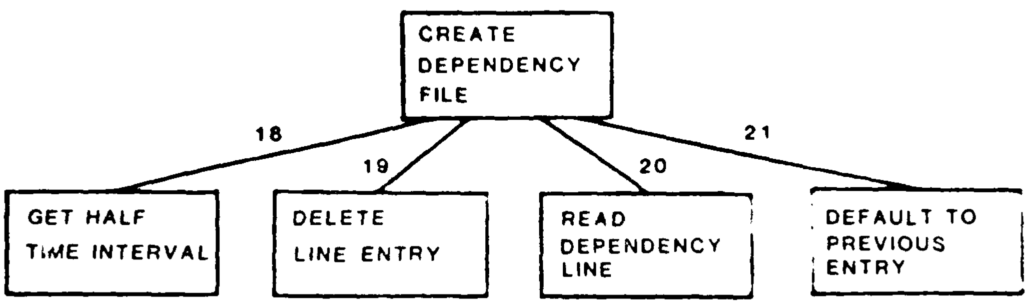
	IN	OUT
8	---	RUN PARAMETERS
9	INITIAL CLOCK SETTING	---
10	CHANNEL SCHEDULE FILE, DEPENDENCY CHECK PARAMETERS, UPDATE READ TIME PARAMETERS, NUMBER OF LOW AND HIGH INPUT SIGNALS	COLLECTED DATA, NUMBER OF COLLECTED DATA, DEPENDENCY CHECK PARAMETERS, POSSIBLE CONTROL CHANNEL PARAMETERS
11	CHANNEL DEPENDENCY FILE, CHANNEL SCHEDULE FILE, UPDATE READ TIME PARAMETERS, DEPENDENCY CHECK PARAMETERS, NUMBER OF LOW AND HIGH INPUT SIGNALS	COLLECTED DATA, NUMBER OF COLLECTED DATA, DEPENDENCY CHECK PARAMETERS, UPDATE READ TIME PARAMETERS
12	CHANNEL CONTROL FILE, POSSIBLE CONTROL CHANNEL PARAMETERS	---
13	RUN IDENTIFICATION, COLLECTED DATA , NUMBER OF COLLECTED DATA	---

Figure 14. Decomposition of "Run Experiment".



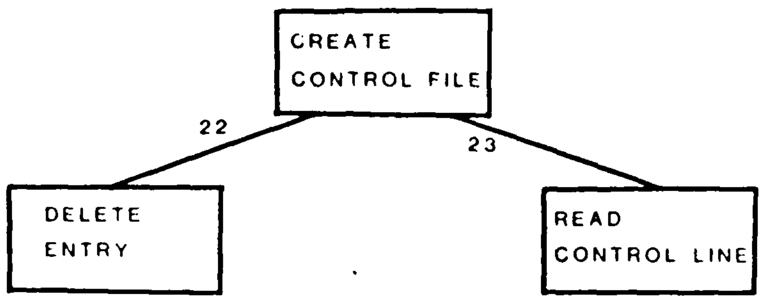
	IN	OUT
14	---	CHANNEL SCHEDULE INFORMATION, NUMBER OF CHARACTERS IN INPUT LINE
15	TIME INTERVAL BETWEEN READS	ONE-HALF OF THE TIME INTERVAL
16	CHANNEL NUMBER, CHANNEL SCHEDULE FILE	UPDATED CHANNEL SCHEDULE FILE
17	READ TIME, TIME BETWEEN READS, CURRENT CHANNEL, PREVIOUS CHANNEL	READ TIME, TIME BETWEEN READS

Figure 15. Decomposition of "Create Schedule File"



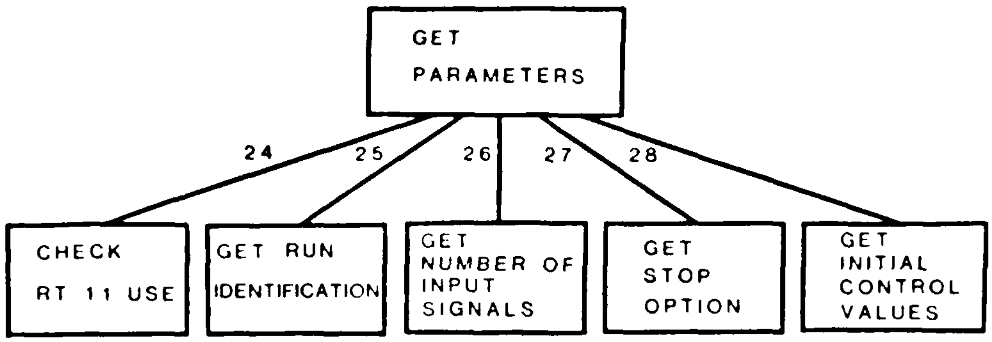
	IN	OUT
18	TIME INTERVAL BETWEEN READS	ONE-HALF OF TIME INTERVAL
19	CHANNEL NUMBER, CHANNEL DEPENDENCY FILE	UPDATED DEPENDENCY FILE
20	---	CHANNEL DEPENDENCY INFORMATION NUMBER OF CHARACTERS IN INPUT LINE
21	TIME BETWEEN READS, CURRENT CHANNEL PREVIOUS CHANNEL	TIME BETWEEN READS

Figure 16. Decomposition of "Create Dependency File"



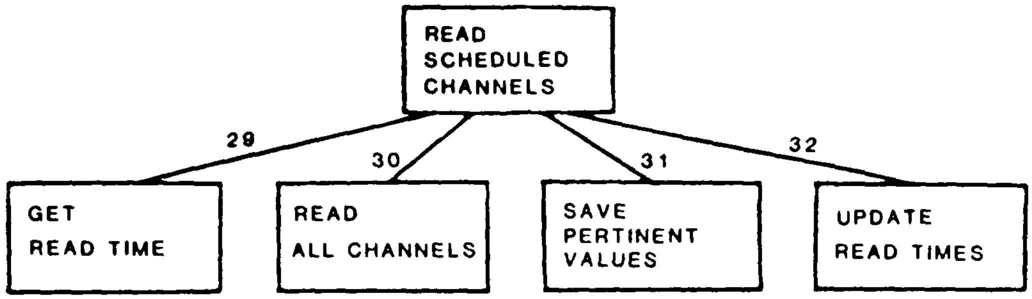
	IN	OUT
22	CHANNEL NUMBER CHANNEL CONTROL FILE	UPDATED CONTROL FILE
23	- - -	CHANNEL CONTROL INFORMATION NUMBER OF CHARACTERS IN INPUT LINE

Figure 17. Decomposition of "Create Control File".



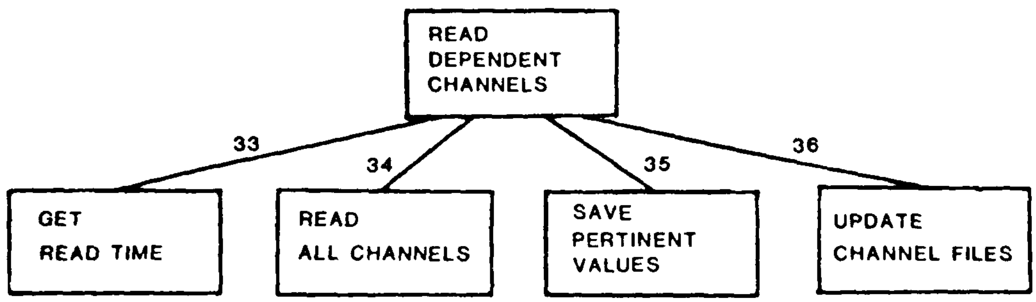
	IN	OUT
24	- - -	- - -
25	- - -	TODAY'S DATE, EXPERIMENT IDENTIFICATION, OPERATOR'S LAST NAME
26	- - -	NUMBER OF LOW INPUT SIGNALS NUMBER OF HIGH INPUT SIGNALS
27	- - -	STOP OPTION (EITHER TIME OR DATA), 4-ELEMENT TIME TO-STOP, TOTAL NUMBER OF DATA POINTS TO BE COLLECTED, NUMBER OF DATA POINTS PER MINICASSETTE DUMP
28	- - -	INITIAL DIGITAL VALUE FOR D/A CHANNEL

Figure 18. Decomposition of "Get Parameters".



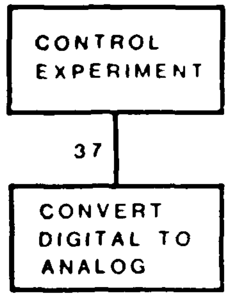
29	---	RECORDED CLOCK TIME
30	NUMBER OF LOW INPUT SIGNALS NUMBER OF HIGH INPUT SIGNALS	CHANNEL VALUES (RAW A/D DATA)
31	NUMBER OF COLLECTED DATA, CHANNEL NUMBER	CHANNEL READ TIME, CHANNEL NUMBER, SCALED CHANNEL VALUE, NUMBER OF COLLECTED DATA
32	READ TIME, CHANNEL NUMBER, NUMBER OF ACTIVE DEPENDENT CHANNELS, NUMBER OF CONTROL CHANNELS, READ TIME OVERRUN INDICATOR	UPDATED CHANNEL READ TIME

Figure 19. Decomposition of "Read Scheduled Channels"



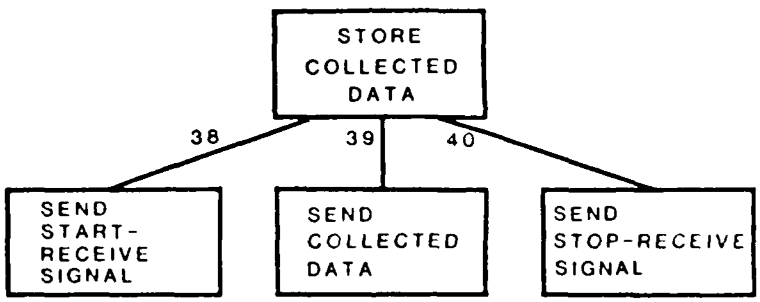
	IN	OUT
33	---	RECORDED CLOCK TIME
34	NUMBER OF LOW INPUT SIGNALS NUMBER OF HIGH INPUT SIGNALS	CHANNEL VALUES (RAW A/D DATA)
35	NUMBER OF COLLECTED DATA, CHANNEL NUMBER	CHANNEL READ TIME, CHANNEL NUMBER SCALED CHANNEL VALUE, NUMBER OF COLLECTED DATA
36	CHANNEL SCHEDULE FILE, CHANNEL NUMBER, STATUS, NUMBER OF ACTIVE DEPENDENT CHANNELS CHANNEL READ TIME	NUMBER OF ACTIVE DEPENDENT CHANNELS, STATUS, UPDATED CHANNEL SCHEDULE FILE

Figure 20. Decomposition of "Read Dependent Channels"



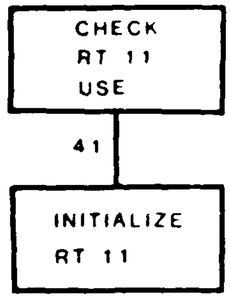
	IN	OUT
37	CHANNEL NUMBER DIGITAL VALUE	- - -

Figure 21. Decomposition of "Control Experiment".



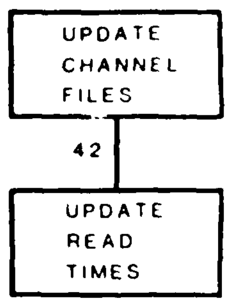
	IN	OUT
38	START-RECEIVE SIGNAL	ACKNOWLEDGE
39	READ TIME, CHANNEL NUMBER, SCALED CHANNEL VALUE NUMBER OF DATA POINTS	- - -
40	STOP-RECEIVE SIGNAL	ACKNOWLEDGE

Figure 22. Decomposition of "Store Collected Data".



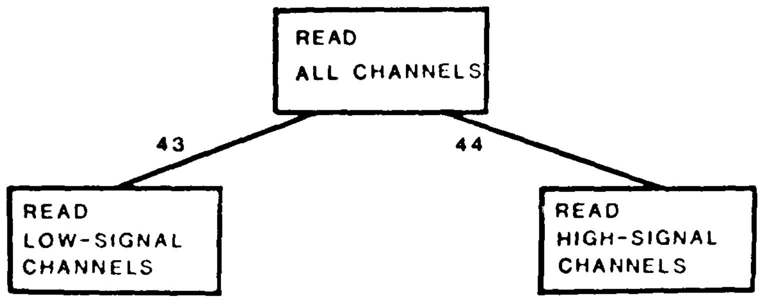
	IN	OUT
41	RT 11 USE INDICATOR	- - -

Figure 23. Decomposition of "Check RT 11 Use"



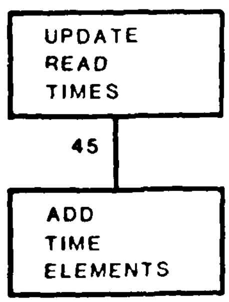
	IN	OUT
42	READ TIME, CHANNEL NUMBER, NUMBER OF ACTIVE DEPENDENT CHANNELS, NUMBER OF CONTROL CHANNELS READ TIME OVER RUN INDICATOR	UPDATED CHANNEL READ TIME

Figure 24. Decomposition of "Update Channel Files"



	IN	OUT
43	NUMBER OF LOW-SIGNAL CHANNELS	RAW CHANNEL VALUES
44	NUMBER OF HIGH-SIGNAL CHANNELS	RAW CHANNEL VALUES

Figure 25. Decomposition of "Read All Channels"



	IN	OUT
45	READ TIME,TIME INTERVAL	READ TIME

Figure 26. Decomposition of "Update Read Times"

Summary

The structure of the software system was revealed in this chapter. The decomposition of the higher-level modules was presented and each module-module interface defined according to the data that flow between them. Against this backdrop, coding the software is made a lot easier, which is, not coincidentally, the next phase in the development effort.

VI. Remaining Development Phases

Introduction

With the structure of the software designed and optimized, it would seem that the next step is to design the logic (algorithm) of the modules and code it in the appropriate programming language. Myers advocates yet another intervening step, however, a step he calls "module interface design". The purpose of module interface design is to produce, for each module, a module interface specification describing precisely the function and interface of the module. The information needed in the specification includes the module (or entry-point) name, the function of the module, the number and order of the arguments, the input arguments (their meaning, format, size, attributes, units, and valid domain), the output arguments (their meaning, format, size, attributes, units, and valid range), and the external effects (actions external to the system, e.g. input/output operations) (Ref 9:148-149).

The module interface specifications provide an excellent documentation mechanism, but are not included here; rather, they will be incorporated in the program code where they are most useful. A listing of the program will be submitted to the Materials Laboratory for inclusion in their application programs library. Future maintenance programmers will find this documentation very valuable and they will have easy access to it.

The remaining phases of the software development for the portable data acquisition system are discussed in this chapter.

Code

Coding the program, given a good structure, becomes almost mechanical (Ref 7:119). This does not mean, however, that the structure be followed rigidly, with no deviations. In fact, the "module interface design" step revealed, and the code implemented, the need to slightly modify the data flow as shown in the design chapter. In particular, the channel schedule, dependency, and control files are no longer passed between modules as arguments but are rather shared by modules through the use of named COMMON blocks. This introduces "pathological data connections" (Ref 19:242) between modules which create, from a structured design point of view, an undesirable coupling condition. However, these channel files each contain a group of functionally-related data items and together total more than a trivial number of arguments to pass from module to module. And although still open for debate, the number of modules these files have to pass through makes the claim credible that this can cause a discernible effect in the speed of the program execution.

The relative ease of coding the program does not inhibit one from using pre-existing modules, either. This is in fact encouraged (Ref 9:4). In the Materials Laboratory, a number of small programs have been written for specific data acquisition systems. The Assembly program to drive the ADAC A/D and D/A converter and the clock interrupt routine are incorporated in the portable data acquisition software. These modules expect COMMON blocks for the values read from the converter and the running clock time information.

Data Collection Algorithm. The top-down approach to the decomposition of the program modules makes it easy to understand what each module performs. The lowest of these modules perform elementary functions and are not difficult to code. The implementation of some higher-level modules, however, is more complex and needs more elucidation. For this purpose, the algorithm for the data collection is presented below.

1. Scheduled channels are read 'regularly' - as closely as possible to the scheduled read times.

a. All channels are continuously scanned through the analog-to-digital converter. The values read, along with the time of this read, are placed in a buffer. A determination is then made whether to save this temporary data or not.

b. If the actual read time is within M1 milliseconds (see point 8 below) before the scheduled read time, the data in the buffer is stored. If it (actual read time) is already past the scheduled read time, the data is also stored, in addition to checking the conditions in 2 below. Otherwise, the data is discarded (overwritten).

2. If the actual read time overruns the scheduled read time, the actual read time is checked to see if it is within one-half the time interval (between scheduled reads) from the scheduled read time. If it is, the scheduled read time is incremented by the corresponding channel time interval to get to the next read time. If it is not, the scheduled read time is incremented by as many time intervals as necessary to get it

within one-half the time interval from the actual read time; the channel values read then correspond to this updated scheduled read time, and normal incrementing is resumed. The skipped scheduled read times are simply missed.

3. If there is a channel scheduled to be read within M2 milliseconds (see point 8 below) of the actual channel read time, no dependency check is made unless as specified in 4 below.

4. If the time interval between scheduled channel reads is such that a defined channel dependency file is not being checked (there is always one channel or another to read within M2 milliseconds of yet another), a mechanism is included to force a dependency check after every fifth actual read and storage of channel data (this condition was approved by the Laboratory sponsor).

5. A dependency check is made whenever time is available or forced by conditions in 4 as long as there is at least one active dependent channel. The set of originally defined dependent channels is scanned and, if active, the value of the corresponding independent channel is tested with the given conditions and values. If the conditions are met, the dependent channel is read, the value is stored, and the proper disposition rule is enforced.

6. Once satisfied, a dependency status may be turned off or let to remain on. When turned off, the dependent channel may be read regularly, if desired, and added to the channel schedule file.

7. A dependency check for each independent channel can be made only between successive actual reads (and storages) for the independent channel, and then only once during this span (i.e., multiple dependency checks between successive channel reads are not allowed).

8. M1 was initially given the value of 10 milliseconds; M2 was given the value of 20 milliseconds. Using the algorithm discussed above, the program was tested with only a channel schedule file defined. The time interval used was 2 milliseconds to force the fastest channel reads possible. (This is not optimum because of point 2 above.) The following results were recorded:

Number of Scheduled Channels	Average Time Between Reads
1	4.25 msec
2	8.00 msec
3	12.00 msec
4	16.10 msec
15	50.20 msec

A channel dependency file (with only one channel defined) was added to the above test. If no dependency check has to be made (point 3), the fastest a scheduled channel can be read is reduced to once every 12 milliseconds. If a dependency check is made but the condition is not satisfied (no dependent channel is read), the scheduled channel read is further slowed down to once every 14 milliseconds. If the condition is met, the rate deteriorates to once every 18 milliseconds.

Given the statistics above, the values of 10 milliseconds for M1 and 20 milliseconds for M2 were retained.

Control Algorithm. As stated earlier, control of the experiment is a rather primitive operation. It is implemented by placing a computed

digital value into the proper input port of the digital-to-analog converter and routing its analog output to the appropriate experiment control equipment. Before the actual start of the experiment, this digital value is initialized by the operator. This analog voltage causes some physical phenomenon to happen, the effect of which is, in turn, measurable by an analog-to-digital converter. The output of the analog-to-digital converter is monitored via a control channel.

For proper control of the experiment, the control channel value must be within a given range (also operator-defined). If it is not, the digital input value to the d/a converter is adjusted up or down depending on which out-of-range condition exists by a certain correction value also initialized by the operator. This digital input value is continually adjusted until such time that the control channel value is within the specified range. If, however, the range is narrow and the control channel value skips from one side (of the range) to the other in one adjustment, the correction value is halved and (finer) adjustments in the opposite direction proceed as before.

Debug

The ease of debugging any computer program depends heavily upon how the code is written. The use of clever, exotic algorithms tend to complicate otherwise simple programs. And when an error is indicated, finding and correcting it can be very frustrating. In the software for the portable data acquisition system, this type of algorithms is avoided. There is no reason for using such algorithms: the modules perform simple, recognizable functions.

Errors do, and did, occur however. But their identification and correction was helped tremendously by means of the debugging tools on the PDP 11/03 development system. The use of the debugging statement indicator (letter D in Fortran statement column 1), and the use of the On-Line Debugging Technique (ODT) were the tools most often utilized in this project. In addition, numerous input data checking statements were added to the code because "debugging input data is as important as debugging a program" (Ref 6:87). Strategically located print statements of specific error messages were also used abundantly throughout the program.

Test and Integration

The top-down decomposition technique has one other very practical feature: it allows the programmer the ability to use "stub modules". These are modules that are written to simulate the functions of subordinate modules. In using this technique, the top module in the program is coded and tested using stub modules for all the modules it calls. Once the top module has been tested, one of its immediate-subordinate modules is coded, the two modules are integrated together along with the necessary stubs, the combination is tested, and so forth (Ref 9:147). This technique worked very well in this project because the particular format selected to implement the system (at least the higher-level portion) is the command format. In this manner, each of the next-to-top level modules is invoked according to which command the user gives interactively with the program. Going further down, each of the channel files is created by the respective module also according to the type interactively specified by the user.

Initial test and integration was performed at the PDP 11/03 development site on those routines not requiring the use of the clock. After this portion of the code had been tested and integrated, the portable data acquisition system itself was used. A dual floppy disk drive, with the RT-11 system floppy disk on one drive and the portable system software on the other, was utilized in this test and integration phase. An HP 3312A Function Generator was used to generate predictable, known signals for input to the analog-to-digital converter.

Operations and Maintenance

A program may be subjected to extensive test and integration procedures but its real test is how it holds up in actual practice. The portable data acquisition system has been used in two different experiments in the Materials Laboratory: high performance liquid chromatography and thermal analysis of polymers. In both of these experiments, only a few-hundred data points were collected. These data points were plotted and compared with strip-chart recordings made concurrently with the data acquisition process. The results show a general resemblance between corresponding plots: the big difference is the presence of intermittent noise spikes on the plots generated by the portable system. The noise problem has been reduced significantly since then; but it still persists.

It should be mentioned that the thermal analysis experiment required only a straightforward data collection capability. The liquid chromatography set-up had two output channels, and they were used as an independent-dependent channel pair. There was no real need to collect

data on very short intervals either, so the data collection algorithm has not really been "pushed" except during the testing phase. The control routines have not yet been exercised on an actual experiment as well. Therefore, there has been no occasion so far to perform what can truly be considered maintenance programming. When the situation arises, the programmer should find ample documentation to make his job easier.

Summary

The remaining phases of the software development were discussed in this chapter. They appear simple and straightforward. But this is only deceptively so. The efforts put forth in the previous phases make the rest of the development job relatively easy.

VII. Summary and Recommendations

The software development effort discussed in this thesis gives one the impression of a smooth, ordered transition from one phase to the next. Actually, there was a significant amount of overlap between them, even a couple of complete iterations from the design of the structure to the writing of the code. This is not normally encouraged; the relative size of this effort made such iteration possible in this case.

The software was designed and coded using structured design techniques which emphasize modularity of components. Higher level modules are logically broken down into smaller, simpler modules. While this technique really does make understanding the software system easier, it does tend to increase the size of the program. For this project, such increase was minimal but did have an impact on the overall scheme of the system. For this size of an effort also, there is a normal tendency to view the emphasis on structured programming techniques as a nuisance; that these techniques make much more impact on larger systems. However, as has already been mentioned, its documentation value alone makes the use of these techniques very worthwhile.

Summary

The stated purpose of this study was to develop the software for the portable, generalized experiment control and data acquisition system. By meticulously following the software development phases recommended by the Air Force, this software product is now operational

and has indeed been used on two Materials Laboratory experiments. The results from these experiments show the collected data with a significant amount of noise. However, this is recognized as a hardware, most probably power supply, problem and does not indicate software insufficiency. Any program errors uncovered during the operational phase of the software life cycle (which this project is in now) will be corrected by the maintenance programmer.

In addition to the power supply problem briefly mentioned above, the project was also delayed considerably during the later stages of the development because of the insufficiency of the portable terminal's minicassette tapes. It was initially expected that any software written for the data acquisition system would fit on one side of a minicassette tape. This may have been the case but for the generality of the program. The use of structured design techniques may also have added to the size of the program. However, it was felt that any drastic cut of the program, in terms of its generality and of its structure, would seriously impact the ease of its use and, ultimately, its maintenance. In addition, a force fit into one minicassette side leaves no room for possible enhancements or modifications. If this room is ever needed later, for any reason at all, then one is back to contending with the same problem.

In an effort to solve this problem, a program was written to "cleanly" break the resulting stand-alone file into two pieces, loading each piece into one side of a minicassette tape. Loading the two pieces of file from the minicassette tape to the portable data acquisition system resulted in numerous loading errors. Using degaussed tapes as

suggested by the portable terminal manufacturer reduced the number of errors; nevertheless, the program load was still unsuccessful. Therefore, the initial system requirement to be able to load the stand-alone program file from the portable terminal's minicassette tapes to the data acquisition system was not satisfied. Currently, the portable data acquisition system can be used only through the use of a dual floppy disk drive, which is the configuration used during the testing and initial operational phases of the development.

Recommendations

Meaningful recommendations for follow-on efforts to the topics considered in this thesis revolve mainly around possible enhancements to the system. First of all, since the automatic control algorithm is admittedly primitive, a more sophisticated mechanism may be incorporated. It must be kept in mind, however, that the system memory size is very limited and care must be taken not to incorporate too extensive an algorithm. Secondly, there is not now any convenient means to abort the program except through the use of the computer RESET button. Pressing this button immediately stops the program and resets the computer. If another experiment run is contemplated, the program must be re-loaded. A different abort technique is definitely in order. Thirdly, the thermal analysis experiment exposed a deficiency in the analog-to-digital conversion method. Previous data collections made with an HP coupler-controller/paper tape arrangement show data beyond the range selected on the thermal analysis equipment. This means that any spikes on the analog input signals are properly and continuously recorded. On the portable data acquisition system, any readings outside

the allowed -10v to +10v range are truncated. Presently, the system can satisfy this continuous recording requirement only by reducing the input signal amplification, thereby losing some resolution. Finally, from the hardware point of view, the noise problem still has to resolved and the limited storage capability of the portable terminal's minicassette tapes is definitely unacceptable. A tape cartridge subsystem (DEC's TU58) with 256K bytes of storage per drive is a viable replacement for the minicassette tapes.

Bibliography

1. An Introduction to Structured Analysis and Design Technique. Waltham, Massachusetts: SofTech, Inc., November 1976.

2. Boehm, Barry W. "Software Engineering," IEEE Transactions on Computers, C-25(12): 1226-1241 (December 1976).

3. DoD Standard 7935.1-S. Automated Data Systems Documentation. Washington: Department of Defense, September 1977.

4. Gardner, Clifford C. An Emitter Data Maintenance and Retrieval System. Thesis, AFIT/GCS/EE/79-5. Dayton, Ohio: Air Force Institute of Technology, December 1979.

5. IDEF, Architect's Manual, ICAM Definition Method (Version 0). Waltham, Massachusetts: SofTech, Inc., September 1979.

6. Kernighan, Brian W. and P. J. Plauger. The Elements of Programming Style (Second Edition). New York: McGraw-Hill Book Company, 1978.

7. Maneely, John R. Design of a Laboratory Data Acquisition System (Time Digitization System). Thesis. AFIT/GCS/EE/78-4. Dayton, Ohio: Air Force Institute of Technology, March, 1978.

8. Martin, James. Design of Man-Computer Dialogues. New Jersey: Prentice-Hall, Inc., 1973.

9. Myers, Glenford J. Composite/Structured Design. New York: Van Nostrand Reinhold Company, 1978.

10. Myers, Ware. "The Need for Software Engineering," Computer, 11(2): 12-24 (February 1978).

11. PDP-11 Fortran Language Reference Manual, RT-11 V3. Maynard, Massachusetts: Digital Equipment Corporation, June 1977.

12. PDP-11 Macro-11 Language Reference Manual. Maynard, Massachusetts: Digital Equipment Corporation, November 1978.

13. Reeve, William H. and Jerry L. Stinson. Software Design for a Visually Coupled Airborne System Simulator (VCASS). Thesis. AFIT/GCS/EE/78-6. AD A055226. Dayton, Ohio: Air Force Institute of Technology, March 1978.

14. Ross, D. T. and K. E. Schoman, Jr. Structured Analysis for Requirements Definition. Waltham, Massachusetts: SofTech, Inc., April 1976.

15. RT-11/RSTS/E Fortran IV User's Guide, RT-11 V03B. Maynard, Massachusetts: Digital Equipment Corporation, June 1977.

16. RT-11 System User's Guide, RT-11 V03B. Maynard, Massachusetts: Digital Equipment Corporation, March 1978.

17. Rutledge, James P. Lecture materials distributed in EE 6.93, Software Engineering. School of Engineering, Air Force Institute of Technology. Wright-Patterson Air Force Base, Spring 1980.

18. Verchot, Edgar A., Jr. Automated Hall Effect Experiment Data Acquisition System (AHEEDAS). Thesis. AFIT/GEO/79D-5. Dayton, Ohio: Air Force Institute of Technology, December 1979.

19. Yourdon, Edward and Larry L. Constantine. Structured Design (Second Edition). New York: Yourdon Press, 1978.

20. Zelkowitz, Marvin V. "Perspectives on Software Engineering," Computing Surveys, 10(2): 197-216 (June 1978).

Appendix A

Sample Program Code

The sample program codes are given below to provide a feel for what the code looks like. In particular, the portion of the code for the main program, GECDAS, is provided to give the reader the details of the channel files mentioned in Chapters III and IV. The format for the introductory comments for each module is that of Myers, specifically the module interface specifications. Other comments, along the same lines, are the author's.

```
      PROGRAM GECDAS
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                      C
C  FUNCTION:  THIS IS THE MAIN PROGRAM FOR THE GENERALIZED EXPERIMENT  C
C             CONTROL AND DATA ACQUISITION SYSTEM (GECDAS).            C
C                                                                      C
C  INTERACTIVE COMMANDS ARE:                                           C
C    DEFINE - DEFINE DATA COLLECTION CHANNELS                          C
C    HELP   - LIST ALL VAID COMMANDS                                   C
C    RUN    - START EXPERIMENT CONTROL AND DATA ACQUISITION            C
C    STOP   - STOP DATA ACQUISITION PROGRAM                            C
C                                                                      C
C......................................................................C
C                                                                      C
C  SUBPROGRAMS CALLED:                                                 C
C    DEFINE - DEFINE DATA COLLECTION CHANNELS                          C
C    HELP   - LIST ALL VALID COMMANDS                                  C
C    RUNEXP - START EXPERIMENT CONTROL AND DATA ACQUISITION            C
C                                                                      C
C......................................................................C
C                                                                      C
C  VARIABLES USED:                                                     C
C    VALID - LOGICAL VARIABLE INDICATING WHETHER USER-GIVEN COMMAND    C
C            IS VALID OR NOT (.TRUE. OR .FALSE.)                       C
C    COMAND - USER-GIVEN (INTERACTIVELY) COMMAND                       C
C                                                                      C
C......................................................................C
C                                                                      C
```

```
C  NAMED COMMON BLOCKS DEFINITION                                          C
C                                                                          C
C    SCDATA (CHANNEL SCHEDULE FILE DATA):                                  C
C      NCHAN - NUMBER OF SCHEDULED CHANNELS                                C
C      ICHAN(I) - THE ITH (SCHEDULED) CHANNEL NUMBER DEFINED               C
C      RTIME(I,J) - SCHEDULED READ TIME FOR ICHAN(I)                       C
C       RTIME(I,1) - HOURS COMPONENT OF CHANNEL READ TIME                  C
C       RTIME(I,2) - MINUTES COMPONENT OF CHANNEL READ TIME                C
C       RTIME(I,3) - SECONDS COMPONENT OF CHANNEL READ TIME                C
C       RTIME(I,4) - MILLISECONDS COMPONENT OF CHANNEL READ TIME           C
C      DTIME(I,J) - TIME INTERVAL BETWEEN CHANNEL READS FOR ICHAN(I)       C
C       DTIME(I,1) - HOURS COMPONENT OF THE TIME INTERVAL                  C
C       DTIME(I,2) - MINUTES COMPONENT OF THE TIME INTERVAL                C
C       DTIME(I,3) - SECONDS COMPONENT OF THE TIME INTERVAL                C
C       DTIME(I,4) - MILLISECONDS COMPONENT OF THE TIME INTERVAL           C
C      HDTIME(I,J) - ONE-HALF OF DTIME(I,J), THE TIME INTERVAL             C
C      SCALE(I) - SCALE FACTOR FOR CHANNEL ICHAN(I) VALUE                  C
C      DNAME(I) - NAME GIVEN TO CHANNEL ICHAN(I)                           C
C                                                                          C
C    DPDATA (CHANNEL DEPENDENCY FILE DATA):                                C
C      NDCH - NUMBER OF DEPENDENT CHANNELS                                 C
C      YCHAN(I) - THE ITH DEPENDENT CHANNEL (NUMBER) DEFINED               C
C      XCHAN(I) - THE INDEPENDENT CHANNEL FOR YCHAN(I); I.E., THE          C
C                 CHANNEL UPON WHOSE VALUE YCHAN(I) DEPENDS FOR READING    C
C      COND1(I) - TOGETHER WITH VALUE1(I), IT MAKES UP THE FIRST           C
C                 CONDITION THA XCHAN(I) MUST MEET FOR YCHAN(I) TO         C
C                 BE READ; CAN HAVE VALUES "GT", "GE", "EQ", "LE", "LT"    C
C      COND2(I),VALUE2(I) - MAKE UP THE SECOND CONDITION, IF APPLICABLE,   C
C                           THAT XCHAN(I) MUST MEET                        C
C      STATUS(I) - INDICATES WHETHER THE DEPENDENCY CONDITION FOR          C
C                  YCHAN(I) IS ACTIVE OR NOT                               C
C      DISP(I) - IF THE DEPENDENCY STATUS FOR YCHAN(I) IS ACTIVE AND       C
C                THE CONDITIONS ARE MET, DISP(I) INDICATES WHETHER TO      C
C                DEACTIVATE THE STATUS OR NOT                              C
C      XCHCHK(I) - A DEPENDENCY CHECK WILL BE MADE ONLY BETWEEN TWO        C
C                  SUCCESSIVE SCHEDULED CHANNEL READS AND THEN ONLY        C
C                  ONCE DURING THIS SPAN;  XCHCHK(I) SIGNIFIES, FOR        C
C                  THE CORRESPONDING XCHAN(I), WHETHER A DEPENDENCY        C
C                  CHECK CAN BE LEGALLY MADE OR NOT                        C
C                                                                          C
C    CNDATA (CHANNEL CONTROL FILE DATA):                                   C
C      NCNTL - NUMBER OF CONTROL CHANNELS                                  C
C      CNTLCH(I) - THE ITH CONTROL CHANNEL (NUMBER) DEFINED                C
C      LOVALU(I) - LOW LIMIT OF ACCEPTABLE CNTLCH(I) VALUES                C
C      HIVALU(I) - HIGH LIMIT OF ACCEPTABLE CNTLCH(I) VALUES               C
C      ACTION(I) - INDICATES WHETHER MANUAL OR AUTOMAIC RESPONSE IS        C
C                  TAKEN WHENEVER THE VALUE OF CNTLCH(I) GETS OUT OF       C
C                  RANGE                                                   C
C      LOCHAN(I) - WHEN CNTLCH(I) VALUE GETS BELOW LOVALU(I) AND           C
C                  ACTION(I) IS AUTOMATIC, LOCHAN(I) IS ADJUSTED           C
C                  ACCORDINGLY                                             C
C      HICHAN(I) - WHEN CNTLCH(I) VALUE IS OVER HIVALU(I) AND ACTION(I)    C
C                  IS AUTOMATIC, ANALOG OUTPUT CHANNEL HICHAN(I) IS        C
C                  ADJUSTED ACCORDINGLY                                    C
```

```
C      OLDREL(I) - THE RELATIONSHIP BETWEEN THE LAST CONTROL CHANNEL      C
C                  VALUE AND ITS ACCEPTABLE VALUES;  HAS ALPHABETIC       C
C                  VALUE "OK" IF VALUE WAS WITHIN RANGE;  "HI" IF         C
C                  LARGER THAN HIVALU(I);  OR "LO" IF SMALLER THAN        C
C                  LOVALU(I)                                              C
C      DIGITL(J) - ADJUSTMENTS ARE MADE TO LOCHAN(I) OR HICHAN(I) (=J)    C
C                  BY LOADING A DIGITAL VALUE (DIGITL(J)) INTO THESE      C
C                  CHANNELS WHICH IS THEN CONVERTED TO THE                C
C                  CORRESPONDING ANALOG VOLTAGE FOR OUTPUT                C
C      DELTA(I) - THE MAGNITUDE OF THE DIGITAL ADJUSTMENT, EITHER         C
C                 ADDED TO OR SUBTRACTED FROM DIGITL(J) ACCORDING TO      C
C                 OLDREL(I)                                               C
C      MESSAG(I,1) - FIRST 8 CHARACTERS OF MESSAGE TO OPERATOR'S          C
C                    TERMINAL WHEN CNTLCH(I) VALUE IS OUT OF RANGE        C
C                    AND ACTION(I) INDICATES MANUAL RESPONSE              C
C      MESSAG(I,2), MESSAG(I,3) - CONTAIN THE REMAINING CHARACTERS OF     C
C                                 THE MESSAGE TO THE OPERATOR             C
C                                                                         C
C    CLKCOM (CURRENT CLOCK INFORMATION)                                   C
C      HR - HOURS COMPONENT OF CURRENT CLOCK TIME                         C
C      MIN - MINUTES COMPONENT OF CURRENT CLOCK TIME                      C
C      SEC - SECONDS COMPONENT OF CURRENT CLOCK TIME                      C
C      MSEC - MILLISECONDS COMPONENT OF CURRENT CLOCK TIME                C
C                                                                         C
C    ADCOM (ANALOG-TO-DIGITAL CONVERTER VALUES)                           C
C      JVALUE - DIGITAL CONVERSION OF ANALOG VOLTAGES PRESENT IN THE      C
C               A/D CHANNELS                                              C
C                                                                         C
C.........................................................................C
C                                                                         C
C  MODULE-MODULE DATA CONNECTION THROUGH COMMON BLOCKS                    C
C                                                                         C
C  :-----------:-----------------------------------------------------:   C
C  :  MODULE   :     N A M E D     C O M M O N     B L O C K       :   C
C  :    NAME   :-----------------------------------------------------:   C
C  :           :  SCDATA : DPDATA :  CNDATA : CLKCOM :  ADCOM       :   C
C  :-----------:---------:--------:---------:--------:--------------:   C
C  :   GECDAS  :    X    :    X    :    X    :    X    :    X      :   C
C  :   HELP    :         :         :         :         :           :   C
C  :   DEFINE  :         :         :         :         :           :   C
C  :   SCHFIL  :    X    :         :         :         :           :   C
C  :   DEPFIL  :         :    X    :         :         :           :   C
C  :   CONFIL  :         :         :    X    :         :           :   C
C  :   RSLINE  :    X    :         :         :         :           :   C
C  :   DVTIME  :         :         :         :         :           :   C
C  :   DELETE  :         :         :         :         :           :   C
C  :   RDLINE  :         :    X    :         :         :           :   C
C  :   DEFALT  :         :         :         :         :           :   C
C  :   RCL[illegible]E  :         :         :    X    :         :           :   C
C  :   RUNEXP  :    X    :    X    :    X    :         :           :   C
C  :   PARAMS  :         :         :         :         :           :   C
C  :   RT11    :         :         :         :         :           :   C
C  :   INITRT  :         :         :         :         :           :   C
C  :   RUNID   :         :         :         :         :           :   C
```

```
C  :   SIGNAL     :         :         :         :         :           :  C
C  :   STOPTN     :         :         :         :         :           :  C
C  :   INITCV     :         :         :    X    :         :           :  C
C  :   SETCLK     :         :         :         :    X    :           :  C
C  :   CLKINT     :         :         :         :    X    :           :  C
C  :   RSCHAN     :    X    :         :         :         :           :  C
C  :   CHKDEP     :         :    X    :         :         :     X     :  C
C  :   UPCHAN     :    X    :         :         :    X    :     X     :  C
C  :   CONEXP     :         :         :    X    :         :           :  C
C  :   DAC        :         :         :         :         :           :  C
C  :   STORE      :         :         :         :         :           :  C
C  :   SNDATA     :         :         :         :         :           :  C
C  :   GETTIM     :         :         :         :    X    :           :  C
C  :   READCH     :         :         :         :         :     X     :  C
C  :   ADC        :         :         :         :         :     X     :  C
C  :   ADAC       :         :         :         :         :           :  C
C  :   SAVE       :    X    :         :         :         :     X     :  C
C  :   UPDATR     :    X    :         :         :         :           :  C
C  :   ADTIME     :         :         :         :         :           :  C
C  :--------------:---------:---------:---------:---------:-----------:  C
C                                                                        C
C***********************************************************************C
C
C
      LOGICAL*1 VALID
      REAL*8    COMAND
C
C
      INTEGER*2 NCHAN,ICHAN(16),RTIME(16,4),DTIME(16,4),HDTIME(16,4)
      REAL*8    DNAME(16)
      REAL*4    SCALE(16)
C
      LOGICAL*1 XCHCHK(16)
      INTEGER*2 NDCH,YCHAN(16),XCHAN(16),COND1(16),COND2(16),
     1          STATUS(16)
      REAL*4    VALUE1(16),VALUE2(16),DISP(16)
C
      INTEGER*2 NCNTL,CNTLCH(16),LOCHAN(16),HICHAN(16),
      INTEGER*2 OLDREL(16),DIGITL(2),DELTA(16)
      REAL*8    ACTION(16),MESSAG(16,3)
      REAL*4    LOVALU(16),HIVALU(16)
C
      INTEGER*2 HR,MIN,SEC,MSEC
C
      INTEGER*2 JVALUE(16)
C
      COMMON /SCDATA/ NCHAN,ICHAN,RTIME,DTIME,HDTIME,SCALE,DNAME
      COMMON /DPDATA/ NDCH,YCHAN,XCHAN,COND1,COND2,DISP,
     1                VALUE1,VALUE2,STATUS,XCHCHK
      COMMON /CNDATA/ NCNTL,CNTLCH,ACTION,LOCHAN,HICHAN,MESSAG,
     1                LOVALU,HIVALU,OLDREL,DIGITL,DELTA
      COMMON /CLKCOM/ HR,MIN,SEC,MSEC
      COMMON /ADCOM/  JVALUE
C
```

```
C
      CALL HELP(VALID)
C
      NCHAN = 0
      NDCH  = 0
      NCNTL = 0
C
C.....GET COMMAND
C
100   WRITE(5,1001)
1001  FORMAT(/2X,'COMMAND? ',$)
C
      READ(5,1002,ERR=200) COMAND
1002  FORMAT(A8)
C
      VALID = .FALSE.
      IF (COMAND.EQ.4HHELP) CALL HELP(VALID)
      IF (COMAND.EQ.6HDEFINE) CALL DEFINE(VALID)
      IF (COMAND.EQ.3HRUN) CALL RUNEXP(VALID)
      IF (COMAND.EQ.STOP) GOTO 900
C
        IF (.NOT.VALID) WRITE(5,1003)
1003    FORMAT(/2X,'INVAID COMMAND; NO COMMAND EXECUTED.')
        GOTO 100
C
200     WRITE(5,2001)
2001    FORMAT(/2X,'*ERROR - INVALID INPUT; TRY AGAIN.')
        GOTO 100
C
900   CONTINUE
C
      STOP
      END
```

```
      SUBROUTINE SNDATA(TODAY,EXPID,OPERTR,RDTIME,CHANEL,VALUE,NDATA)
C
C*******************************************************************C
C                                                                   C
C  FUNCTION:  WRITE RUN INFO AND COLLECTED DATA ONTO MINICASSETTE   C
C                                                                   C
C  CALLING SEQUENCE: CALL SNDATA(TODAY,EXPID,OPERTR,                C
C                              RDTIME,CHANEL,VALUE,NDATA)           C
C                                                                   C
C  INPUTS:                                                          C
C    TODAY  - TODAY'S DATE                                          C
C    EXPID  - EXPERIMENT IDENTIFICATION                             C
C    OPERTR - OPERATOR'S LAST NAME                                  C
C    RDTIME - CHANNEL READ TIME                                     C
C    CHANEL - CHANNEL NUMBER                                        C
C    VALUE  - SCALED CHANNEL VALUE                                  C
C    NDATA  - NUMBER OF DATA POINTS                                 C
C                                                                   C
C  OUTPUTS:  NONE                                                   C
C                                                                   C
C  EXTERNAL EFFECTS:                                                C
C    COLLECTED DATA ARE WRITTEN ONTO THE TERMINAL MINICASSETTE      C
C                                                                   C
C...................................................................C
C                                                                   C
C  CALLED BY:                                                       C
C    STORE - STORE DATA IN TERMINAL'S MINICASSETTE                  C
C                                                                   C
C  SUBPROGRAMS CALLED:  NONE                                        C
C                                                                   C
C  VARIABLES USED:                                                  C
C    TODAY  - TODAY'S DATE                                          C
C    EXPID  - EXPERIMENT IDENTIFICATION                             C
C    OPERTR - OPERATOR'S LAST NAME                                  C
C    RDTIME - RECORDED CHANNEL READ TIME                            C
C    CHANEL - CHANNEL NUMBER                                        C
C    VALUE  - SCALED CHANNEL VALUE                                  C
C    NDATA  - NUMBER OF DATA POINTS TO WRITE ONTO MINICASSETTE      C
C                                                                   C
C*******************************************************************C
C
      INTEGER*2 RDTIME(1175,4),CHANEL(1175),NDATA
      REAL*4    VALUE(1175)
      REAL*8    TODAY,EXPID(2),OPERTR
C
C.....WRITE RUN IDENTIFICATION
C
      WRITE(5,1001) EXPID(1),EXPID(2)
1001  FORMAT(/2X,'EXPERIMENT: ',2A8)
C
      WRITE(5,1002) OPERTR,TODAY
1002  FORMAT(2X,'OPERATOR: ',A8,'      DATE: ',A8)
C
C.....WRITE DATA ONTO MINICASSETTE
```

```
C
      WRITE(5,1003)
1003  FORMAT(/2X,'  H M S M CH  VALUE')
      DO 200 I=1,NDATA
         WRITE(5,1004) (RDTIME(I,J),J=1,4),CHANEL(I),VALUE(I)
1004     FORMAT(1X,I4,2I2,I3,I2,F8.2)
200   CONTINUE
C
      RETURN
      END
```

Appendix B

Users Manual

Introduction

The objective of the Users Manual is to provide non-ADP personnel with the information necessary to effectively use the system (Ref). However, because of the interactive nature of the input requirements for the program, the Users Manual for the portable data acquisition software is very much simplified. As shown in the next sections, the Users Manual is nothing more than a list of suggestions on how to get the most out of the system.

Input Formats

Input formats are made explicit by system prompts and headings. An "A" format requires an alphabetic entry; an "N", a numeric entry; and an "X", an alphabetic or a numeric entry. Line entries are titled so that required information may be typed underneath the respective heading. For example (user entries are underlined):

```
 CH    CHANNEL     SCALE     START READ TIME      TIME INTERVAL
  #    DATA NAME   FACTOR    HR  MIN SEC MSEC    HR  MIN SEC MSEC
 NN    XXXXXXXX    NNNN.NN   NN  NN  NN  NNN     NN  NN  NN  NNN
> 1    CH1           1.                                   5
> 2    CH2
> 3    CH3
> 4    CH4
>
```

This line entry results in (see Appendix C for more examples):

CH	CHANNEL	SCALE	START READ TIME				TIME INTERVAL			
#	DATA NAME	FACTOR	HR	MIN	SEC	MSEC	HR	MIN	SEC	MSEC
1	CH1	1.00	0	0	0	0	0	0	5	0
2	CH2	1.00	0	0	0	0	0	0	5	0
3	CH3	1.00	0	0	0	0	0	0	5	0
4	CH4	1.00	0	0	0	0	0	0	5	0

Here, one of the most powerful features of the system is revealed. After defining the first channel fully, the user need not do the same for the other channels. The redundant information about the start read time and the time interval is omitted. By defaulting to the values of the previous channel, the system automatically assigns the same values to the channels down the line. This automatic default to the previous channel's value occurs only, however, for values in an incomplete entry line. For example:

CH	CHANNEL	SCALE	START READ TIME				TIME INTERVAL			
#	DATA NAME	FACTOR	HR	MIN	SEC	MSEC	HR	MIN	SEC	MSEC
NN	XXXXXXXX	NNNN.NN	NN	NN	NN	NNN	NN	NN	NN	NNN
> 1	CH1	1.0		1	1					500
> 2	CH2	1.0							2	
> 3		2.0			5					
>										

would result in:

CH	CHANNEL	SCALE	START READ TIME				TIME INTERVAL			
#	DATA NAME	FACTOR	HR	MIN	SEC	MSEC	HR	MIN	SEC	MSEC
1	CH1	1.00	0	1	1	0	0	0	0	500
2	CH2	1.00	0	0	0	0	0	0	2	0
3		2.00	0	0	5	0	0	0	2	0

Other user entries are not as complicated and, as already stated, properly prompted by the system. In addition, the system echoes all pertinent input values to give the user a chance to double-check his information before proceeding. For inputs requiring specific entries, a list of possible valid entries is provided. The program will not proceed to the next executable statement unless one of the valid entries is received.

Output Format

The output of the system is the collected data dumped into the terminal minicassette tape. Each time a dump to the minicassette is indicated, the proper run identification is placed in front of the data. This run information includes the experiment identification, the date, and the operator's last name. The data is written in the format hhhhmmsskkknnxxxxx.xx; where hhhhmmsskkk refers to the time the data was recorded in hours (hhhh), minutes (mm), seconds (ss), and milliseconds (kkk); nn is the channel number; and xxxxx.xx is the scaled channel value.

Channel Numbering

The portable data acquisition system accepts, as inputs, both low-level (millivolts range) and high-level (-10v to +10v) signals from loboratory equipments. The high-level signals are connected directly to the analog-to-digital converter. The low-level signals are multiplexed and amplified before they are fed to the converter. The output of the amplifier is permanently connected to channel 0 (defined as channel 1 in the program) of the converter. Therefore, no high-level signal can be defined as channel 1 in the program.

If there are more than one low-level signals, the channels are numbered 1 to N, where N is the number of low-level signals to be monitored. High-level signals, if present, are then numbered N+1 to N+M, where M is the number of high-level signals. If only high-level

signals are to be monitored, they must be numbered 2 to M+1.

Channel Naming

Provision was made in the program for the capability to measure rates (of change per unit time). To do this, the user simply uses the word "RATE" as the first four letters of the particular channel's data name. The value stored for this channel is properly computed to depict its change in value per second. Heat up rates and rate of crack growth are typical measurements being made in the Laboratory.

General Hints

Scale factors can be used to good advantage by realizing that the analog-to-digital input range of -10 volts to +10 volts produces a digital conversion of -2048 to 2047. The digital value multiplied by 4.88 gives the signal level in millivolts.

If, for some reason or another, a need arises to monitor a certain signal with two channels, the user is reminded that it is perfectly legal to do so. Using two different, judiciously selected scale factors, one can monitor actual values and rates at the same time.

Appendix C

Sample Session

The following "printout" is transcribed from a sample session at the operator's terminal after all the necessary channel connections between the portable data acquisition system and the experiment equipments have been made. The underlined words or letters are user entries. Small letters enclosed in parentheses are provided in some instances for clarification.

```
THE FOLLOWING IS A LIST OF VALID COMMANDS

    HELP     - LIST ALL VALID COMMANDS
    DEFINE   - DEFINE CHANNELS TO BE USED
               (NORMALLY, THE FIRST COMMAND GIVEN)
    RUN      - START DATA ACQUISITION PROCESS
               (GIVEN AFTER ALL CHANNELS ARE DEFINED)
    STOP     - STOP THE PROGRAM

COMMAND?  DEFINE
(S)CHEDULE, (D)EPENDENCY, OR (C)ONTROL FILE? OR (Q)UIT? S

 CH    CHANNEL     SCALE     START READ TIME       TIME INTERVAL
  #   DATA NAME   FACTOR    HR  MIN  SEC  MSEC   HR  MIN  SEC  MSEC
 NN   XXXXXXXX   NNNN.NN    NN   NN   NN  NNN    NN   NN   NN   NNN
> 1      CH1        1.0                                    5
> 2      CH2
> 3      CH3
> 4      CH4
>(here, the operator hits the carriage return key immediately.)

THE FOLLOWING INFORMATION WAS SUPPLIED:

 CH    CHANNEL     SCALE     START READ TIME       TIME INTERVAL
  #   DATA NAME   FACTOR    HR  MIN  SEC  MSEC   HR  MIN  SEC  MSEC
  1      CH1        1.00     0    0    0    0     0    0    5    0
  2      CH2        1.00     0    0    0    0     0    0    5    0
  3      CH3        1.00     0    0    0    0     0    0    5    0
  4      CH4        1.00     0    0    0    0     0    0    5    0

ANY CHANGES (Y/N)? Y
CHANGE ALL (CA), LOCAL (CL), ADD (AD), DELETE (DL), OR (Q)UIT? DL

CHANNEL? 4

CHANNEL? 1
```

```
CHANNEL? (immediate carriage return)

THE FOLLOWING INFORMATION WAS SUPPLIED:

 CH    CHANNEL     SCALE     START READ TIME      TIME INTERVAL
  #   DATA NAME    FACTOR    HR  MIN  SEC MSEC    HR  MIN  SEC MSEC
  3      CH3        1.00      0   0    0    0      0   0    5    0
  2      CH2        1.00      0   0    0    0      0   0    5    0

ANY CHANGES (Y/N)? Y
CHANGE ALL (CA), LOCAL (CL), ADD (AD), DELETE (DL), OR (Q)UIT? CL

 CH    CHANNEL     SCALE     START READ TIME      TIME INTERVAL
  #   DATA NAME    FACTOR    HR  MIN  SEC MSEC    HR  MIN  SEC MSEC
 NN   XXXXXXXX    NNNN.NN    NN  NN   NN  NNN     NN  NN   NN  NNN
> 2      CH2        1.0                1                       100
>(immediate carriage return)

THE FOLLOWING INFORMATION WAS SUPPLIED:

 CH    CHANNEL     SCALE     START READ TIME      TIME INTERVAL
  #   DATA NAME    FACTOR    HR  MIN  SEC MSEC    HR  MIN  SEC MSEC
  3      CH3        1.00      0   0    0    0      0   0    5    0
  2      CH2        1.00      0   0    1    0      0   0    0  100

ANY CHANGES (Y/N)? Y
CHANGE ALL (CA), LOCAL (CL), ADD (AD), DELETE (DL), OR (Q)UIT? AD

 CH    CHANNEL     SCALE     START READ TIME      TIME INTERVAL
  #   DATA NAME    FACTOR    HR  MIN  SEC MSEC    HR  MIN  SEC MSEC
 NN   XXXXXXXX    NNNN.NN    NN  NN   NN  NNN     NN  NN   NN  NNN
> 1      CH1
>(immediate carriage return)

THE FOLLOWING INFORMATION WAS SUPPLIED:

 CH    CHANNEL     SCALE     START READ TIME      TIME INTERVAL
  #   DATA NAME    FACTOR    HR  MIN  SEC MSEC    HR  MIN  SEC MSEC
  3      CH3        1.00      0   0    0    0      0   0    5    0
  2      CH2        1.00      0   0    1    0      0   0    0  100
  1      CH1        1.00      0   0    1    0      0   0    0  100

ANY CHANGES (Y/N)? Y
CHANGE ALL (CA), LOCAL (CL), ADD (AD), DELETE (DL), OR (Q)UIT? CA

 CH    CHANNEL     SCALE     START READ TIME      TIME INTERVAL
  #   DATA NAME    FACTOR    HR  MIN  SEC MSEC    HR  MIN  SEC MSEC
 NN   XXXXXXXX    NNNN.NN    NN  NN   NN  NNN     NN  NN   NN  NNN
> 1     TEMP1A      4.88               1                   1   500
> 2     TEMP1B      4.88               2                   2
> 3     TEMP1C      4.88     24
> 4     TEMP1D      4.88     48
> 5     LENGTH      1.0          1                         10
```

> 6 RATE 6.0
>(immediate carriage return)

THE FOLLOWING INFORMATION WAS SUPPLIED:

CH	CHANNEL	SCALE	START READ TIME				TIME INTERVAL			
#	DATA NAME	FACTOR	HR	MIN	SEC	MSEC	HR	MIN	SEC	MSEC
1	TEMP1A	4.88	0	0	1	0	0	0	1	500
2	TEMP1B	4.88	0	0	2	0	0	0	2	0
3	TEMP1C	4.88	24	0	2	0	0	0	2	0
4	TEMP1D	4.88	48	0	2	0	0	0	2	0
5	LENGTH	1.00	0	1	0	0	0	0	10	0
6	RATE	6.00	0	1	0	0	0	0	10	0

ANY CHANGES (Y/N)? N

COMMAND? DEFINE

(S)CHEDULE, (D)EPENDENCY, OR (C)ONTROL FILE? OR (Q)UIT? D

DEP	CHANNEL	SCALE	IND	COND		COND			TIME INTERVAL			
CH	DATA NAME	FACTOR	CH	1	VALUE1	2	VALUE2	DISP	HR	MIN	SEC	MSEC
NN	XXXXXXXX	NNNN.NN	NN	AA	NNNN.NN	AA	NNNN.NN	AAA	NN	NN	NN	NNN
> 7	TEMP2	4.88	1	GT	500.00	LT	525.00	ON				
> 8	TEMP3	4.88	3	GE	1200.00	LE	1212.00	OFF	0	0	1	500

>(immediate carriage return)

THE FOLLOWING INFORMATION WAS SUPPLIED:

DEP	CHANNEL	SCALE	IND	COND		COND			TIME INTERVAL			
CH	DATA NAME	FACTOR	CH	1	VALUE1	2	VALUE2	DISP	HR	MIN	SEC	MSEC
7	TEMP2	4.88	1	GT	500.00	LT	525.00	ON				
8	TEMP3	4.88	3	GE	1200.00	LE	1212.00	OFF	0	0	1	500

ANY CHANGES (Y/N)? Y
CHANGE ALL (CA), LOCAL (CL), ADD (AD), DELETE (DL), OR (Q)UIT ? Q

ANY CHANGES (Y/N)? N

COMMAND? DEFINE

(S)CHEDULE, (D)EPENDENCY, OR (C)ONTROL FILE? OR (Q)UIT? C

CTL	CONTROL CHANNEL		ACTION IF	OUTPUT CH		IF OPERATOR ACTION REQUIRED,
CH	OK IF WITHIN		OUT OF RANGE	IF CTL CH		ENTER MESSAGE TO OPERATOR
#	VALUE1 TO	VALUE2	(OPER/OUTPUT)	LOW	HIGH	(OPER TYPES "C" TO CONTINUE)
NN	NNNN.NN -	NNNN.NN	AAAAA	NN	NN	AAAAAAAAAAAAAAAAAAAAAAAA
> 4	350.00	360.00	OPER			CHANNEL 1 OUT OF RANGE!!
> 6	35.00	3.5	OUTPUT	1	2	

>(immediate carriage return)

THE FOLLOWING INFORMATION WAS SUPPLIED:

CTL CH #	CONTROL CHANNEL OK IF WITHIN VALUE1 TO VALUE2		ACTION IF OUT OF RANGE (OPER/OUTPUT)	OUTPUT CH IF CTL CH LOW	HIGH	IF OPERATOR ACTION REQUIRED, ENTER MESSAGE TO OPERATOR (OPER TYPES "C" TO CONTINUE)
4	350.00	360.00	OPER			CHANNEL 1 OUT OF RANGE!!
6	35.00	3.50	OUTPUT	1	2	

ANY CHANGES (Y/N)? N

COMMAND? RUN

ARE YOU RUNNING RT-11? (Y/N)...Y

ENTER TODAY'S DATE (DDMMMYY, 20FEB81)...22FEB81

ENTER EXPERIMENT ID (16 CHAR MAXIMUM)...MECHANICAL TEST

ENTER OPERATOR LAST NAME (8 CHAR MAXIMUM)...ILORETA

ENTER NUMBER OF SIGNALS TO BE AMPLIFIED (MAX, 16): NMUX = 5
IS 5 OK? (Y/N)...N

ENTER NUMBER OF SIGNALS TO BE AMPLIFIED (MAX, 16): NUMX = 4
IS 4 OK? (Y/N)...Y

ENTER NUMBER OF SIGNALS THAT NEED NO AMPLIFICATION (<16-NMUX): 2
IS 2 OK? (Y/N)...Y

ENTER STOP OPTION (DATA OR TIME) : DATA

ENTER NUMBER OF DATA POINTS TO BE COLLECTED: 3000
IS 3000 OK? (Y/N)...Y

ENTER NUMBER OF DATA POINTS PER DUMP TO THE MINICASSETTE (MAX, 1100): 1100
IS 1100 OK? (Y/N)...N

ENTER NUMBER OF DATA POINTS PER DUMP TO THE MINICASSETTE (MAX, 1100): 500
IS 500 OK? (Y/N)...Y

***MAKE SURE MINICASSETTE IS SET FOR WRITING!!!
***TURN THE SYSTEM CLOCK ON!!!

ENTER 'GO' TO START THE DATA ACQUISITION PROCESS: GO

As soon as the operator hits the return key, the data gathering process is started. At this point, he may turn the printer on his terminal OFF. For this particular example session, however, he should not because of his possible interaction when a control channel gets out of range. Assuming the experiment runs to completion and the printer is ON when a dump to the

minicassette tape is indicated, the following header and example data may be printed:

```
EXPERIMENT: MECHANICAL TEST
OPERATOR: ILORETA        DATE: 22FEB81

  H M S M CH  VALUE
  0 0 0991 1  537.52
  0 0 1991 2  323.48
  0 0 2491 1  525.74
  0 0 3991 1  520.85
  0 0 3991 2  330.95
  0 0 4   5 1  518.26
  0 0 4   5 7  152.47
```

VITA

Arsenio R. Iloreta was born on 14 December 1949 in Ilocos Sur, Philippines. He graduated from high school in Ilocos Sur in 1964. He then attended the University of the Philippines, from which he received a Bachelor of Science in Chemical Engineering in 1969. He was a research assistant at the University of the Philippines before coming to the United States in 1971. In 1972, he enlisted in the United States Air Force. He was working as an electronic computer system repairman at Offutt Air Force Base, Nebraska, when he was selected to go to the Officers' Training School in 1976. His next assignment was as a materials research engineer at Wright-Patterson Air Force Base, Ohio. His interest in the computer science field earned him a position as a computer system development officer at the Materials Laboratory. He has been a part-time student at the Air Force Institute of Technology for four years.

Permanent Address: 1884 Hulali Loop
Puu Kaa Subdivision
Kapaa, Hawaii 96746

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: AFIT/GCS/EE/81M-3

2. GOVT ACCESSION NO.: AD-A100 [illegible]

3. RECIPIENT'S CATALOG NUMBER:

4. TITLE (and Subtitle): A GENERALIZED EXPERIMENT CONTROL AND DATA ACQUISITION SYSTEM

5. TYPE OF REPORT & PERIOD COVERED: MS Thesis

6. PERFORMING ORG. REPORT NUMBER:

7. AUTHOR(s):
ARSENIO R. ILORETA
Captain USAF

8. CONTRACT OR GRANT NUMBER(s):

9. PERFORMING ORGANIZATION NAME AND ADDRESS:
Air Force Institute of Technology (AFIT/EN)
Wright-Patterson AFB, Ohio 45433

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS:

11. CONTROLLING OFFICE NAME AND ADDRESS:
Air Force Wright Aeronautical Laboratories
Materials Laboratory (AFWAL/MLU)
Materials Technical Services Division
Wright-Patterson AFB, Ohio 45433

12. REPORT DATE: March 1981

13. NUMBER OF PAGES: 93

14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office):

15. SECURITY CLASS. (of this report): Unclassified

15a. DECLASSIFICATION DOWNGRADING SCHEDULE:

16. DISTRIBUTION STATEMENT (of this Report):
Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report):

18. SUPPLEMENTARY NOTES:
Approved for public release; IAW AFR 190-17.
FREDRIC C. LYNCH, Major, USAF
Director of Public Affairs

19. KEY WORDS (Continue on reverse side if necessary and identify by block number):
Software Engineering, Structured Design, Data Acquisition
Experiment Control, LSI-11 Microcomputer

20. ABSTRACT (Continue on reverse side if necessary and identify by block number):

A portable, automated, generalized experiment control and data acquisition system was developed for the Materials Laboratory to fulfill some of its data automation needs. The system is portable in that all hardware components fit in an aluminum carrying case, except for the terminal wich is separate. The need for portability requires the program to be loaded from the terminal minicassette tape. The collected data is likewise [illegible] into the minicassette.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

The Air Force method of requirements definition, IDEF0, was used to model the software requirements. It involves a top-down approach to development. The model, thus diagrammed, provided the general structure from which the program itself was written. Structured code and ample documentation form the basic programming technique implemented. The program is currently operational, the system having been utilized in two actual experiment set-ups. The requirement to load the program from the minicassette was not satisfied, however, because of some loader error. The program can presently be loaded using a floppy disk system.

END